Technical University of Munich

Department of Informatics

Master's Thesis in Informatics

Stochastic Block Models for Analysis and Synthetic Generation of Communication Networks

Patrick Kalmbach

Technical University of Munich

Department of Informatics

Master's Thesis in Informatics

Stochastic Block Models for Analysis and Synthetic Generation of Communication Networks

Stochastische Blockmodelle für die Analyse und synthetische Generierung von Kommunikationsnetzen

Author:	Patrick Kalmbach
Advisor:	Prof. Dr.-Ing. Wolfgang Kellerer
Supervisors:	Andreas Blenk and Markus Klügel
Submission Date:	15.05.2107

With my signature below, I assert that the work in this thesis has been composed by myself independently and no source materials or aids other than those mentioned in the thesis have been used.

München, 03.04.2017	
Place, Date	Signature

München, 03.04.2017	
Place, Date	Signature

Abstract

Communication Networks grow ever more complex. The increasing complexity poses a challenge for network operators and researchers alike. To perform network management tasks, e.g., network security or resource allocation, or design new algorithms, an understanding of the underlying structure of communication networks is indispensable. Furthermore, researchers require realistic synthetic graphs to validate new algorithms.

This thesis uses probabilistic models to infer structure in network data represented as graphs, and uses the inferred models to generate synthetic ones. The parameters of the models can be inferred from empirical data and encode the structure of the graph, which is thus learned in an unsupervised fashion. The models used in this thesis are the Stochastic Block Model (SBM) and Degree Corrected Block Model (DCBM), a variant allowing for heavy tailed degree distributions.

The SBM and DCBM are applied to different graphs extracted from IP-to-IP communication of an enterprise network. The resulting groups are analyzed with respect to contained network services and topological roles of vertices, i.e., client and server roles. The evaluation shows that both models separate vertices primarily according to their topological role. The structure inferred with the SBM is clearer and better interpretable than that of the DCBM.

Generative capabilities of the models are evaluated by generating synthetic graphs for an IP-to-IP communication-, and router level graph. The quality of generated graphs is asserted by evaluating different graph features, and comparing the results with the state of the art generator Orbis. The results show that SBM and DCBM are on par with, or even outperform Orbis.

To showcase the capabilities of the SBM for network management tasks, botnet detection in IP-to-IP communication of a university network is performed. The proposed approach achieves a true positive rate of 100 % once the malware becomes active, and a false negative rate of 2.5 %.

Contents

Chapter 1

Introduction

Communication networks have grown ever more complex over the past decades. Be it the Internet graph on the router or AS-level, or IP-to-IP communication networks in campus, enterprise or backbone networks. The complexity poses challenges for network operators and researchers alike. For network managers, challenges are, e.g., resource allocation, network security or network design [TPGK03]. Challenges for researchers include the design of topology aware algorithms [AGL15]. An understanding of the underlying structure of communication networks is indispensable to successfully tackle the mentioned challenges. Furthermore, researchers are in need for realistic synthetic graphs, e.g., for the development of new algorithms and applications. This thesis applies Latent Variable Models (LVMs) to assist operators and researchers when addressing the aforementioned challenges. LVMs represent a parametric probability distribution and can be used for compression and generation of relational data, e.g., data of communication networks. This thesis understands communication networks as graphs, i.e., as relational data, and uses two LVMs for graph data, namely the Stochastic Block Model (SBM) and Degree Corrected Block Model (DCBM), to obtain compressed representations of communication networks. Furthermore, being LVMs, SBM and DCBM can generate synthetic graphs.

The SBM has already been introduced in the 1980s [HLL83] in the social sciences. Since then, the topic has established itself and experienced a lot of attention from other fields [GZFA10]. For example, social sciences and biology rely on the SBM for community detection or identification of functional groups in protein-protein interaction networks [KN11, AJC15, NC15, Pei14].

The SBM models a graph as a set of different groups. Groups can be, e.g., communities of friends in social networks. Each vertex belongs to exactly one group. Edges form solely based on the membership of incident vertices. The group membership and edge probabilities between groups form a parametric probability distribution. The parameters can be estimated from empirical data using

principled statistical methods. Once the model parameters are known, samples, i.e., graphs in the case of the SBM, can be drawn from the resulting probability distribution. The SBM thus goes beyond being a simple tool for network management and operation, and can additionally be used as synthetic graph generator. To the best of our knowledge, none of the previous work is able to provide similar functionality, i.e., to infer structure and to use it for generation.

Synthetic graph generation is an important problem in its own right. Synthetic graphs are used to develop, test and verify, e.g., routing or resource management algorithms [RFT13]. Many models and generators that produce graphs with properties similar to communication networks, mainly the Internet topology, have been proposed. Examples are preferential attachment models or generators [Zho06, Kea15, MLMB01] replicating the degree distribution [MKFV06, MHK+07] or matching structural patterns [AGL15, CDZ97, HZRR15]. However, many network generators replicate only a subset of the original graphs' properties [AGL15, RFT13]. In addition, communication networks change over time or develop new characteristics, which makes a continuous and extensive analysis necessary. Other graph types, e.g., IP-to-IP communication between hosts, which is used for anomaly detection, resource management or traffic classification, are important [IGER+10, AKM+05, JGL15]. A generator modeling the current Internet topology might fail to generate realistic graphs in this case. Most of the existing network generators cannot generate synthetic graphs for arbitrary domains due to their underlying assumptions. In summary, synthetic graph generation is difficult due to the extensive knowledge necessary to identify and model patterns, and the dynamic and heterogeneous nature of graphs from different types of communication networks.

In contrast, the SBM does not make explicit assumptions about the underlying structure of the graph, and has thus the potential to generate synthetic graphs for different domains. By using the SBM, extensive graph analysis, necessary for synthetic graph generation today [AGL15], is replaced by inference of model parameters from relational data, e.g., from traffic traces. Given the SBM, synthetic graphs for different application domains can be generated by virtually anyone. No deep understanding of graph theory is necessary, and no strong assumptions are made by the SBM regarding the structure of graphs, contrasting it from other approaches for synthetic graph generation. The capabilities to generate graphs for different communication networks is shown in this thesis.

The goals of this thesis are thus threefold. Firstly, this thesis investigates the capabilities of the SBM to infer meaningful groups in different types of communication networks, namely IP-to-IP communication networks and router level topologies. Secondly, this thesis investigates the capabilities of the SBM to generate realistic synthetic graphs from empirical samples. Thirdly, the usefulness of the

inferred groups in IP-to-IP communication networks is showcased by identifying hosts being part of a botnet in a university network.

This thesis provides the following contributions:

- Application of two probabilistic generative models and their directed variants to communication networks, namely the SBM and DCBM.
- A detailed evaluation of the capabilities of the models to infer meaningful groups with respect to logical and topological roles of vertices in different graph types arising in the context of communication networks.
- Evaluation of the applicability of the models to the task of synthetic graph generation for different types of networks arising in the context of communication networks.
- Proof of concept how the SBM can be used to detect anomalous hosts in networks by identifying hosts being part of a botnet in a university network.

The remainder of this thesis is organized as follows; Chapter 2 gives concepts and definitions necessary for the understanding of the remainder of this thesis. In particular, Chapter 2 introduces the used probabilistic models. Chapter 3 introduces the related work with respect to analysis an generation of groups an anomaly detection. Chapter 4 analyzes the capabilities of the SBM and DCBM to infer meaningful groups in graphs based on IP communication. Chapter 5 turns towards synthetic graph generation and analyzes the capabilities of SBM and DCBM to infer meaningful topological structure and generate realistic synthetic graphs from examples. Chapter 6 provides a proof of concept how groups inferred with the SBM can be used for security in a campus network. Chapter 7 closes the thesis by giving a summary and providing possible directions for future research.

Chapter 2

Background

This chapter introduces terms and techniques used throughout this thesis and being essential for its understanding. The methodology presented in Section 2.2.1 allows a concise and easily interpretable representation of the probabilistic models introduced in Section 2.4. Section 2.1 introduces basic concepts required for the statistical analysis of graphs. Section 2.3 introduces basic concepts, taxonomy and techniques for anomaly detection that become important in Chapter 6.

2.1 Graph Theory

This section introduces basics in graph theory that are essential for the following chapters. Section 2.1.1 introduces different graph types and how they are represented. Graph and vertex level attributes used in this work are introduced in Section 2.1.2 and Section 2.1.3 introduces different generative models for graphs, that are heavily used in engineering and statistical physics. The models used in this thesis are related to models introduced in this section and share some of their properties. Thus, an understanding of the basic models given in this section aids in understanding some of the results in later chapters.

2.1.1 Graph Types

A graph $G = (\mathcal{V}, \mathcal{E})$ consists of vertices $\mathcal{V} = \{v_1, \ldots, v_N\}$ and edges $\mathcal{E}$. A graph can be represented as a matrix called adjacency matrix, which is denoted as A. The entries A_{ij} give the number of edges that run from vertex v_i to vertex v_j. The nature of $\mathcal{E}$ gives raise to different types of graphs:

- Simple directed graphs,
- Simple undirected graphs,

- directed Multi-Graphs,
- undirected Multi-Graphs,

that are represented differently.

Simple directed graphs Edges have a direction and are a subset of the crossproduct of vertices. More formally:

$$\mathcal{E} := \{(v_i, v_j) \subset \mathcal{V} \times \mathcal{V} \mid v_i \neq v_j\}. \tag{2.1}$$

The adjacency matrix A is an a-symmetric binary matrix: $A \in \{0,1\}^{N \times N}$ and $A \neq A^T$. The vertex an edge originates at is called *tail*, and the vertex an edge terminates at is called head. The row-sums of A give the out-degree, and the column-sums the in-degree of each vertex. The out-degree is the number of edges that originate from a vertex, and consequently the in-degree is the number of edges that terminate at a vertex. The degree of each vertex in a directed graph is the sum of in- and out-degree. The sum over all entries in A consequently gives exactly the number of edges.

Simple undirected graph Can be viewed as a directed graph, containing for every outgoing edge a respective incoming edge: $(v_i, v_j) \in \mathcal{E} \Leftrightarrow (v_j, v_i) \in \mathcal{E}$. As a result, the adjacency matrix is a symmetric, binary matrix: $A \in \{0,1\}^{N \times N}$ and $A = A^T$. This type of graph is often referred to as *simple graph*. Both, row- and column-sums give the degree of each vertex. Contrary to the simple directed graph results the sum over A in **twice** the number of edges.

Directed multi-graph Edges are a multi-set over the crossproduct of vertices. Each edge can appear multiple times. A multi-graph may also contain self edges, that are excluded in the simple directed and undirected case. Consequently, the adjacency matrix A is a-symmetric and contains natural numbers: $A \in \mathbb{N}_0^{N \times N}$. Each cell A_{ij} gives the number of edges running from vertex v_i to vertex v_j, and diagonal elements A_{ii} give the number of self-edges. The in- and out-degree are analogue to the simple directed graph. As in the case of the simple directed graph results the sum over A in exactly the number of edges of the multi-graph.

Undirected multi-graph Analogous to the simple graphs, the set of undirected multi-graphs is a subset of the directed multi-graphs. Let the function $m(\cdot)$ give the number of times an element appears in a multi-set: $m : \mathcal{V} \times \mathcal{V} \longleftarrow \mathbb{N}_0$. Then $m((v_i, v_j)) = m((v_j, v_i))$. The adjacency matrix is a symmetric matrix containing natural numbers: $A \in \mathbb{N}_0^{N \times N}$ and $A = A^T$. In contrast to the directed multi-graph give the diagonal elements A_{ii} **twice** the number of self-edges, and the sum over A gives **twice** the number of edges in the multi-graph.

In communication networks all graph types occure. For example the graph of the Internet at the AS-level or the router level is represented as a simple undirected graph. IP-to-IP communication graphs can be viewed as either directed or undirected (one host communicates with another but not vice verca). Multi-graphs can be used to represent the number of times one host contacted another or the number of packages send. Also, multi-graphs simplify mathematical treatment of models defined on them. A fact that is used in Section 2.4.

2.1.2 Graph and vertex-level Attributes

Once the structure of a graph is known, different attributes can be calculated. Those attributes allow the identification of different graph types, which in turn allow conclusions about e.g. the behavior of routing algorithms. The most obvious attributes are graph level attributes. They map a graph to a scalar value, and are thus very coarse as many (different) graphs can share them. vertex level attribute are more fine grained, as one value is calculated for each vertex. This thesis defines vertex and graph level attributes as functions. Upper case letters symbolize graph attributes, and lower case letters indicate vertex level attributes. The previous section introduced such an attribute, the degree of each vertex. It is easy to see, that the set of graphs having the same average degree is larger than the set of graphs, in which exactly the same degrees occur for each vertex. Quite a view vertex-level attributes deal with centrality, that is importance, of the vertex in the graph. *Importance* is of course application dependent and thus many different centrality measures exist.

The remainder of this section introduces graph- and vertex-level attributes used throughout this thesis.

Number of vertices of a graph $G = (\mathcal{V}, \mathcal{E})$ is defined as the cardinality $\mid \mathcal{V} \mid$ of the set of vertices $\mathcal{V}$.

Number of Edges of a graph $G = (\mathcal{V}, \mathcal{E})$ is defined as the cardinality $\mid \mathcal{E} \mid$ of the set of edges $\mathcal{E}$.

A Path between two vertices $v, w \in \mathcal{V}$ in a graph $G = (\mathcal{V}, \mathcal{E})$ is a set of vertices $p(v, w) := \{v_1, v_2, \ldots, v_n \mid v_1 = v \wedge v_n = w\}$. The length of a path is given by the cardinality $\mid p(v, w) \mid$. The set $\mathcal{P}_{uv} := \{p(u, v) \subset \mathcal{V} \mid\mid p(u, v) \mid \text{ is minimal}\}$ is the set of shortest paths between vertices u, v. By convention, the distance $m(u, w)$ between two vertices u, w is the length of any shortest path between the two vertices. If no path exists between vertices u, w, then $\mid p(u, w) \mid := \infty$.

Degree Centrality of a vertex v in a graph G is simply the degree $d(v)$ of v:

$$d(v) := \begin{cases} \frac{1}{2} \mid \{w \in \mathcal{V} \mid (v,w) \in \mathcal{E}\} \cup \{w \in \mathcal{V} \mid (w,v) \in \mathcal{E}\} \mid & \text{if } G \text{ undirected} \\ \mid \{w \in \mathcal{V} \mid (v,w) \in \mathcal{E}\} \cup \{w \in \mathcal{V} \mid (w,v) \in \mathcal{E}\} \mid & \text{else.} \end{cases} \tag{2.2}$$

For directed graphs, the out degree $d^{out}(v)$ and in degree $d^{in}(v)$ is just the number of edges originating or terminating at vertex v respectively.

Average Degree of a graph G is the average over all vertex degrees:

$$D(G) := \frac{1}{\mid \mathcal{V} \mid} \sum_{v \in \mathcal{V}} d(v). \tag{2.3}$$

Closeness Centrality measures the mean distance from a vertex to all others and is usually defined as:

$$c(v_i) := \frac{\mid \mathcal{V} \mid -1}{\sum_{v_j \in \mathcal{V}/v_i} m(v_i, v_j)}, \tag{2.4}$$

where $m(\cdot,\cdot)$ is the distance between two vertices. Small values of the closeness centrality indicate a short distance of the respective vertex to all others. Therefore, the placement of content or management utilities on such vertices might be of advantage, as the vertices are easily reachable [New10].

The closeness centrality as defined in Equation (2.4) is very unstable as pointed out in [New10]. Small changes in the underlying graph might cause huge changes in the ordering of the vertices, as the range closeness values fall into is quite small. Thus [New10] proposes an alternative definition:

$$c'(v_i) := \frac{1}{\mid \mathcal{V} \mid -1} \sum_{v_j \in \mathcal{V}/v_i} \frac{1}{m(v_i, v_j)}, \tag{2.5}$$

which does not have this issue and also comes with a few other nice properties. Thus, the definition of centrality in Equation 2.5 is used throughout this thesis.

Betweenness Centrality measures the extend to which a vertex lies in the paths between other vertices. A measure like this can be used in the context of communication networks to identify congestion points or place traffic monitoring devices. Let $r(u,v,w) :=\mid \{p(v,w) \in \mathcal{P}_{vw} : u \in p(v,w)\} \mid$ be the number of times vertex u appears in the shortest paths between vertices v, w, and let $\mid \mathcal{P}_{vw} \mid$ be

the total number of shortest paths between two vertices. Then the betweenness centrality of a vertex v_i is defined as:

$$b(v_i) := \sum_{v_s, v_t \in \mathcal{V}_{\neg i} \times \mathcal{V}_{\neg i}} \frac{r(v_i, v_s, v_t)}{\mid \mathcal{P}_{v_s v_t} \mid}, \tag{2.6}$$

where $\mathcal{V}_{\neg i}$ is short for $\mathcal{V}/v_i$ [New10].

Density of a directed graph is the fraction of realized edges over possible edges:

$$T(G) := \begin{cases} \frac{2|\mathcal{E}|}{|\mathcal{V}|(|\mathcal{V}|-1)} & \text{if } G \text{ is simple and undirected} \\ \frac{|\mathcal{E}|}{|\mathcal{V}|(|\mathcal{V}|-1)} & \text{if } G \text{ is simple and directed} \\ \frac{2|\mathcal{E}|}{|\mathcal{V}|^2} & \text{if } G \text{ is an undirected multi-graph} \\ \frac{|\mathcal{E}|}{|\mathcal{V}|^2} & \text{if } G \text{ is a directed multi-graph} \end{cases} \tag{2.7}$$

For simple directed and undirected graphs $T(G) \in [0, 1]$ while for multi-graphs $T(G) \in [0, \infty)$. Density is a global measure and has some drawbacks. For example, a sparse graph with one dense component might have the same density as a more regularly connected graph.

Cliques and k-plexes are an indicator for highly connected subgroups. A clique is a maximal subset of vertices, such that every vertex is connected to all other vertices. That is, all edges are present and the density within the clique is one.

A relaxation to the strong definition of cliques are k-plexes. A k-plex of size n is a maximally connected subgraph S of a graph G, such that each vertex in S is connected to at least $n - k$ other vertices in S.

A vertex can be a member in multiple k-plexes and cliques at the same time [New10].

Coreness (Centrality) is based on the *k-core decomposition* of a network [Sei83]. A k-core is a maximal connected subgraph $S = (\mathcal{V}', \mathcal{E}')$ of a graph $G = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$, in which all vertices have at least degree k. The term *maximal connected* refers in this context to the existence a vertex $v \in \mathcal{V}$ and $v \notin \mathcal{V}'$ having a degree $d_v \geq k$ that can be added to $\mathcal{V}'$ and S remains connected. Since there might be multiple such components for some k, the biggest component (thus *maximal*) is taken. A vertex has coreness k, if it belongs to the k-core but not the $k+1$-core. The coreness of a graph is the largest coreness score of a vertex [New10].

The concept of the k-core was originally introduced as a global measure of the structure of a graph, differentiating between strongly connected regions and

regions which are not [Sei83]. From this point of view, coreness can be seen as an indicator of hierarchical structure in the network. Coreness can also be interpreted as a measure of robustness against vertex removal. Graphs with a high coreness are more robust. As a centrality for each vertex, coreness indicates the hierarchical level a vertex belongs to, and thus how important that vertex is [GXX15]. Contrary to cliques and k-plexes, k-cores are not overlapping [New10].

Clustering Coefficient is a way to measure transitivity in a graph. For graphs the concept of transitivity can be understood as: if the edges (u, v) and (v, w) exist, then the edge (u, w) also exists. Thus, the clustering coefficient C of a graph can also be understood as an indicator for the formation of communities, if C is larger as expected compared to a random graph [New10]. The following definition is used in this thesis:

$$L(G) := \frac{1}{\mid \mathcal{V} \mid} \sum_{v \in \mathcal{V}} \frac{2tri(v)}{d(v)\,(d(v)-1)}, \tag{2.8}$$

where $tri : \mathcal{V} \to \mathbb{N}$ gives the number of triangles a vertex belongs to [HSS08].

Average Shortest Path Length is for a graph $G = (\mathcal{V}, \mathcal{E})$, the average length over the shortest paths between all pairs of vertices [New10]. More specifically:

$$\bar{m} := \frac{1}{\mid \mathcal{V} \mid (\mid \mathcal{V} \mid -1)} \sum_{(u,v) \in \mathcal{V} \times \mathcal{V}} m(u, v), \tag{2.9}$$

This function is not defined in general for un-connected graphs, i.e. it is infinity. Therefore, the average shortest path length is calculated for each connected component of the un-connected graph G separately.

The average shortest path length allows is an indicator whether a graph has the small world property or not, i.e. a small average shortest path indicates the small world property, and many real world graphs express this structure [New10]

Average Degree Connectivity of a graph $G = (\mathcal{V}, \mathcal{E})$ is the average neighbor degree of vertices with a specific degree. Let $\mathcal{V}_k := \{v \in \mathcal{V} \mid d(v) = k\}$ be the set of vertices with degree k, then the average degree connectivity is defined as:

$$ADC(G, k) := \frac{1}{(\mid \mathcal{V}_k \mid -1)} \sum_{u \in \mathcal{V}_k} \left(\frac{1}{\sum_{j \in \mathcal{V}} A_{uv}} \sum_{v \in \mathcal{V}} A_{uv} d(v) \right). \tag{2.10}$$

The average degree connectivity allows the identification of assortative or dissortative properties, when compared to the average degree of G. If the average over

all ADC values for every degree k occurring in G is larger than the average degree of G, then high degree vertices tend to connect to vertices with lower degree. If G is weighted and directed, then comparing the average degree connectivity values for the weighted and the unweighted adjacency matrix of G indicate the correlation between edge weight and neighbor degree. That is, let ADG and ADG^w be the average degree connectivity values for the unweighted and weighted adjacency matrix of G. If $ADG < ADG^w$, then edges with larger weight point to neighbors with larger degree. Respectively, if $ADG < ADG^w$ then edges with smaller weight point to neighbors with smaller degree. Therefore, ADG^w measures the effective affinity to connect with high or low degree neighbors given the magnitude of the interaction [BBPSV04].

Average Neighbor Degree for a graph $G = (\mathcal{V}, \mathcal{E})$ is related to the average degree connectivity and defined as:

$$ANG(G) := \frac{1}{\mid \mathcal{V} \mid} \sum_{u \in \mathcal{V}} \left(\frac{1}{\mid \mathcal{N}_u \mid} \sum_{v \in \mathcal{N}_u} d(v) \right), \tag{2.11}$$

where $\mathcal{N}_u := \{v \in \mathcal{V} \mid (u, v) \in \mathcal{E} \vee (v, u) \in \mathcal{E}\}$ is the set of adjacent vertices to u. This measure also indicates assortativity or dissortativity with respect to the vertex degree when comparing it with the average degree of G.

Degree Assortativity Coefficient Is the Pearson correlation coefficient between the fraction of edges that leave a vertex with degree x and enter a vertex with another degree y. Let $\mathcal{D} := \{d(v) \mid v \in \mathcal{V}\}$ be the set of all degrees in G, and $\mathcal{E}_{xy} := \{(v, w) \in \mathcal{E} \mid d(v) = x \wedge d(w) = y\}$ be the set of all edges originating at a vertex with degree x and terminating at a vertex with degree y. The the degree assortativity coefficient is defined as [New03]:

$$r(G) := \frac{\sum_{x,y \in \mathcal{D}} \frac{xy}{|\mathcal{E}|} \left(\mathcal{E}_{xy} - \mathcal{E}_{xx} \mathcal{E}_{yy} \right)}{\sigma_{xx} \sigma_{yy}}, \tag{2.12}$$

where σ_{xx} is the standard deviation over $\{\frac{|\mathcal{E}_{xx}|}{|\mathcal{E}|} \mid x \in \mathcal{D}\}$. The degree assortativity coefficient takes values in $r \in [-1, 1]$, where $r = 1$ indicates perfect assortativity and $r = -1$ perfect disassortativity, i.e. Vertices with the same degree and vertices with different degree connect respectively [New03]. The degree assortativity coefficient is a special case, the same formalism can be applied to any scalar vertex attribute.

2.1.3 Graph Models

This Section describe some of the best known models for generating synthetic networks that are used in statistics and engineering. Also, the probabilistic models being introduced in Section 2.4 can be understood as mixtures of some of the models presented in this section.

Erdös Rényi Random Graph or simply called random graph and referred to as $G(n, p)$, where n is the number of vertices and p the probability of edges between vertices. Thus $G(n, p)$ is an ensemble of graphs, and each graph $G = (\mathcal{V}, \mathcal{E})$ exits with probability [New10]:

$$P(G) = p^{|\mathcal{E}|}(1-p)^{\binom{|\mathcal{V}|}{2}-|\mathcal{E}|}. \tag{2.13}$$

A graph is drawn from the ensemble by considering all possible edges $(u, v) \in \mathcal{V}\times\mathcal{V}$, and flipping a coin for each edge with probability of success p, i.e., each edge exists with probability $\text{Ber}(p)$, where $\text{Ber}(\cdot)$ is the Bernoulli distribution.

Some of the graph features introduced in Section 2.1 can be calculated analytically for the $G(n, p)$ model. For example the average number of edges across the ensemble is [New10]:

$$\langle \mid \mathcal{E} \mid \rangle = \sum_{|\mathcal{E}|=0}^{\binom{|\mathcal{V}|}{2}} \mid \mathcal{E} \mid P(\mid \mathcal{E} \mid) = \binom{\mid \mathcal{V} \mid}{2} p, \tag{2.14}$$

which is just the mean of a Binomial distribution. It is easy to see why this is the case. For each edge a Bernoulli experiment, in total $\binom{|\mathcal{V}|}{2}$ experiments, is performed. The number of successes, i.e. the number of existing edges is then distributed according to a binomial distribution. Similarly, the mean degree over the ensemble (i.e., **not** the expected degree of a single graph) can be calculated as:

$$\langle d \rangle = (\mid \mathcal{V} \mid -1)p, \tag{2.15}$$

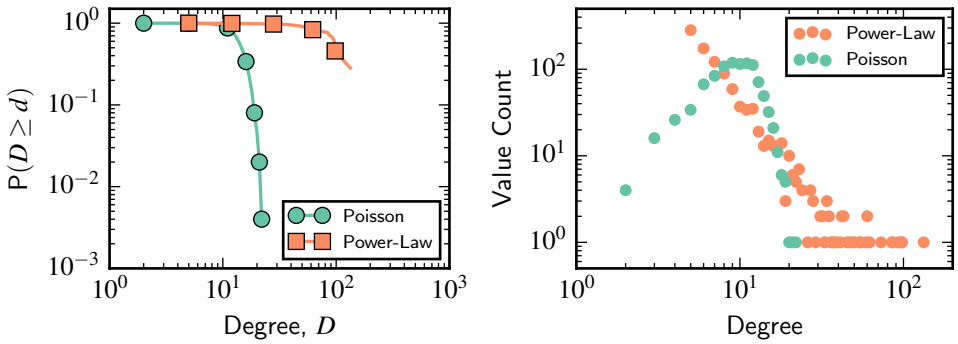
using the fact of the average degree being $\bar{d} = \frac{2}{n} \mid \mathcal{E} \mid$ for such a graph. In a similar fashion, other graph feature can be calculated for the ensemble. Table 2.1 gives an overview. For a detailed derivation see [New10].

Although very popular, the $G(n, p)$ model has several sever drawbacks:

- The Clustering Coefficient is very small opposed to many empirical networks (especially in social sciences, less so for communication networks).
- No correlation between degree of adjacent vertices.
- No community structure.

Degree Distribution	Clustering Coefficient	Giant Component	Diameter	Reciprocity
$Poi(\langle k \rangle)$	$O\left(\frac{1}{\|\mathcal{V}\|}\right)$	for $\langle k \rangle > 1$	$O\left(\log \mid \mathcal{V} \mid\right)$	$O\left(\frac{1}{\|\mathcal{V}\|}\right)$

Table 2.1: Graph properties for the $G(n, p)$ model.



(a) CCDF of degree distribution. (b) Scatter plot how often degree occured

Figure 2.1: CCDF of a graph with poisson and power-law degree distribution. Bot graphs have 1 000 vertices and an average degree of 10.

- Most importantly: Degree distribution follows a Poisson distribution as opposed to a power-law distribution observed in most real world networks.

Especially the Poisson degree distribution is a serious drawback, as this has an impact on many properties observed in real world graphs. With the $G(n, p)$ model it is very unlikely to generate a graph with high degree vertices (heavy tail of the degree distribution) that occur frequently in empirical networks. Figure 2.1 shows the complementary cumulative distribution function (CCDF) and a scatter plot of a graph generated with the $G(n, p)$ model having a Poisson degree distribution and another graph having a power-law distribution. Both graphs have 1 000 vertices and an average degree of about 10. Figure 2.1a shows the CCDF and illustrates the differences between the degree distributions regarding the probability of observing specific degrees. For the graph generated with the $G(n, p)$ model the probability of having a degree that is larger than 20 is around 0.004, while for the graph with power-law distribution this probability is almost 1. Even vertices with large degrees over 100 exist with probability larger than 0.1. The probability values are reflected in Figure 2.1b. For the $G(n, p)$ model the distribution of the value counts for specific degrees takes on a bell shaped curve, while for the graph with power-law degree distribution the values follow more a straight line. Consequently, the $G(n, p)$ model is not suited to generate for example communication networks

Degree Distribution	Clustering Coefficient	Giant Component	Diameter	Reciprocity
specified	$O\left(\frac{1}{\|\mathcal{V}\|}\right)$	for $\langle k^2\rangle - 2\langle k\rangle > 1$	$O\left(\log \mid \mathcal{V} \mid\right) O\left(\frac{1}{\|\mathcal{V}\|}\right)$	

Table 2.2: Graph properties for the $G(n, p)$ model.

with server and clients, as the model is not possible to generate the necessary heterogeneous degree distribution. This property of the $G(n, p)$ model becomes important for the synthesis of graphs in Chapter 5.

Configuration Model The configuration model addresses the problem of the Poisson degree distribution and allows the generation of a random graph with arbitrary degree distribution. To achieve this, the model takes as input a degree sequence $d_1, \ldots, d_n$, i.e., the degree of each vertex $v_1, \ldots, v_n$, and the degree sequence must sum to an even number, i.e. $\sum_{i=1}^{n} d_i \in \{2k \mid k \in \mathbb{N}\}$. With this condition in mind, the degrees can be drawn from any possible distribution. A graph is then generated by adding d_i stubs to each vertex v_i. An edge is formed by choosing two stubs uniformly at random and connecting the two vertices. The result is a particular matching of the stubs and every matching has the same probability. Now it is also clear why the degrees must sum to an even number. If they do not, the resulting graph does not have the specified degree distribution has stubs are left over. The configuration model is thus an ensemble of graphs created by any possible matching of the stubs, and having exactly the specified degree distribution (graphs with other degree distribution have probability zero) [New10].

The configuration model has a few catches. It creates undirected multi-graphs, i.e. the resulting graphs may contain self-loops and multi-edges. The fact that each matching has the same probability does not result in a uniform distribution over graphs, as multiple matchings might have the same graph, i.e., graphs with the same topology, as a result. Thus some graphs of the ensemble have higher probability than others [New10].

Table 2.2 shows some properties for graphs generated with the configuration model. Aside from the degree distribution, they are very similar to the $G(n, p)$ model.

Barabaási-Albert Model or preferential attachment model was developed with evolving graphs in mind. The goal of the model is to simulate a graph, in which vertices are added over time to aid the analysis of dynamic graphs such as the World-Wide-Web or citation networks and possibly project them into future. Each time a vertex is added to the graph, it connects to exactly $a \in \mathbb{N}$

Degree Distribution	Clustering Coefficient	Giant Component	Diameter	Reciprocity
$P(d) \sim d^{-3}$	$\frac{\log^2\lvert\mathcal{V}\rvert}{\lvert\mathcal{V}\rvert}$	always	$O\left(\frac{\log\lvert\mathcal{V}\rvert}{\log\log\lvert\mathcal{V}\rvert}\right)$	NA

Table 2.3: Graph properties for the Barabási-Albert model.

already existing vertices. Each edge forms with probability proportional to the degree of the existing vertex [New10]. The resulting graph has a power-law degree distribution with exponent exactly $\gamma = 3$ [Cla]. The resulting graph has a core-periphery like structure, where the "oldest" vertices are clustered in the middle, surrounded by vertices with lower degree. The central, high-degree vertices also have high betweenness and closeness centrality scores. Moreover, the graph is also degree disassortative, i.e. Vertices of different degree (high and low) are connected with each other. Table 2.3 gives some properties of graphs generated with this model [New10, Cla, KE02]. The graphs generated with the Barabási-Albert model match many characteristics observed empirically in (dynamic) graphs. As a result, extensions have been proposed. For example to generate graphs following a power-law with arbitrary exponent γ of incorporate the small world effect [KE02].

The Waxman Model is a random graph model explicitly incorporating spatial location of vertices, and is an alternative to the $G(n,p)$ model producing more realistic graphs for communication networks. Vertices are placed at random in a unit square and are then independently connected based on their distance, with probability [Wax88]:

$$p((v_i, v_j)) := \beta \exp(-\frac{d(v_i, v_j)}{\beta d_{max}}). \tag{2.16}$$

Here $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is the euclidean distance between vertices v_i and v_j and d_{max} is the maximum distance between any pair of vertices. The parameters $\alpha, \beta \in [0, 1)$ are both related to the formation of edges. Parameter β controls the density, i.e. larger values for β result in a larger density. Parameter α controls the ratio of short edges to long edges. Small values of α increase the number of short edges relative to long ones and vice verca [Wax88].

Obviously, the concept behind the Waxman model is not restricted to a unit square. In fact, any metric space can be used for vertex placement. Since the edges are formed based on the respective metric, properties of graphs generated with the Waxman model depend on the distribution of distance between vertices. Therefore, analytical expressions as given for previous models are quite involved or do not exist. Some expressions are derived in [RTP15] but are too involved

to simply state them here. That being said, [MMB00] empirically found graphs generated with the Waxman model do not express a power-law degree distribution.

2.2 Probability and Information Theory

In this thesis a probabilistic view on graphs is taken. The models being introduced in Section 2.4 define a parametric probability distribution over possible graphs, i.e., graphs are assumed to be generated from a stochastic process. This section introduces basic concepts of both, probability and information theory necessary for the understanding of the used models.

2.2.1 Graphical Models

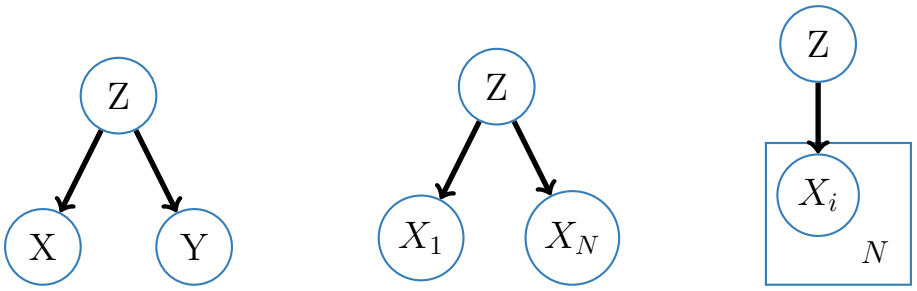
A Graphical Model (GM) is a technique to compactly represent complicated, multivariate probability distributions as a graph. Formulas for such distributions are usually very complex or do not even exist in closed forms. Using GMs has several advantages:

- Concise representation of large distributions
- GM can be used for inference, i.e. as model of the system (here graph) to answer questions
- allows efficient construction of the model, both by a human expert, or automatically by learning from data.

In Section 2.2.2 the general concept of GMs is introduced, and Section 2.2.3 introduces a more specific one, which forms the basis of unsupervised learning [Mur, Chap. 10, 11], [KF, Chap. 1], and is used to represents the models introduced in Section 2.4.

2.2.2 Directed Graphical Models

A Directed Graphical Model (DGM), the most simple form of GMs, represents any joint probability distribution in form of a Directed Acyclic Graph (DAG) $G = (\mathcal{V}, \mathcal{E})$. A joint probability distribution is a powerful tool. If it is known, influences of different RVs in the distribution on each other can be assessed, as well as marginal distribution for single RVs calculated. Also, if a joint probability distribution is fully specified, i.e., all parameters are known, samples can be drawn from the distribution, which will be used in Chapter 5. A node in the DAG represents a Random Variable (RV) and each edge a Conditional Independence



(a) CI statement (b) Without Plate Notation (c) With Plate Notation

Figure 2.2: Left, representation of the CI statement in Equation 2.17. Middle CI statements where many variables have one parent variable. Right, the same statement using plate notation.

(CI) statement. Two RVs X and Y are conditionally independent given a third RV Z iff:

$$X \perp Y \mid Z \iff p(X, Y \mid Z) = p(X \mid Z)p(Y \mid Z). \tag{2.17}$$

The CI statement in Equation (2.17) is represented in Figure 2.2. In a DAG that represents a probability distribution, each node $v_i \in \mathcal{V}$ is conditionally independent of all preceding nodes given the parent nodes [Mur]:

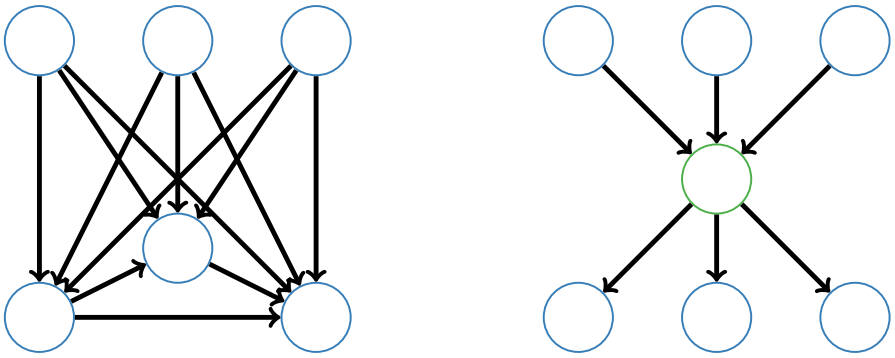
$$v_i \perp pred(v_i)/pa(v_i) \mid pa(v_i), \tag{2.18}$$

where $pred(v_i)$ gives the predecessors, and $pa(v_i)$ the parents of node v_i. This is called the *ordered Markov property* and is one of the key properties of DAGs. The ordered Markov property greatly reduces the number of parameters necessary to describe the model. This property plays in important role when deriving parameter estimations for the parametric models used in this thesis.

To compactly represent DAGs in which many vertices have the same parent the plate notation is used. A plate is a rectangle in which part of the DAG can be placed. A number in the plate specifies how often the enclosed part is repeated. Figure 2.2c shows an example. Here the variables X_i all depend on the variables Z. The number in the lower right of the rectangle specifies the number of repetitions of the part it contains [Mur, Chap. 10]. Compared to Figure 2.2b, the plate notation improves readability and makes the dependence explicit.

2.2.3 Latent Variable Model

A DGM models correlations in the visible space, that is, dependencies between observable RVs. However, the observed correlations might be caused by an unobserved variable, a hidden common cause. Such variables are called hidden or



(a) DGM (b) LVM

Figure 2.3: Graphical Model with- and without common cause. The left models has 59 parameter, whereas the right model has only 17 parameter.

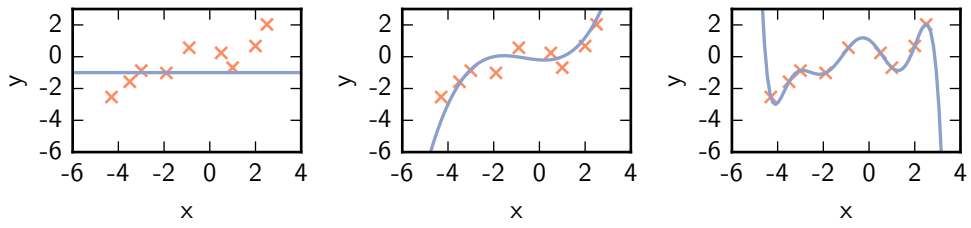
latent variables and a respective model is called Latent Variable Model (LVM). LVMs have two advantages over models modeling correlations in the visible space only:

- LVMs have fewer parameters compared to models in the visible space, and can thus be estimated easier from observed data (see Figure 2.3).
- By reducing observations to a common cause, LVMs result in a compact representation.

On the downside, they are harder to fit than models without latent variables. A well known example for a LVM is the Gaussian Mixture Model (GMM), which can be viewed as a generalization of the K-Means algorithm [Mur, Chap. 11]. LVMs are important for this thesis, as the models being introduced in Section 2.4 are LVMs for graphs.

2.2.4 Model Selection

Model selection addresses the problem of choosing a model that describes a given data set well. Why can't one simply take the model that yields the smallest error? Figure 2.4 illustrates this. Figure 2.4 shows a data set with three possible functions fit to it. Figure 2.4a shows a polynomial of degree zero, i.e. a constant function. This function does not fit the data well. To better adapt to the data, the model can be made more complex. Figure 2.4b shows the fit of a polynomial of degree three. This approximation explains the data quite well, but is off at several points. Figure 2.4c shows the fit of a polynomial of degree eight. This function perfectly fits the data and has zero error, and is therefore superior over the other models.



(a) Degree zero (b) Degree three (c) Degree eight

Figure 2.4: Simple vs. Complex Function. Fit of functions with different degrees to the same data points.

However, the data points are in fact generated by adding Gaussian white noise[1] to the function values of the cubic function in Figure 2.4b. It is easy to see how the complex model will fail to generalize to unseen data. This highlight the difference between fitting to the data and actually modeling the underlying pattern. On the other hand, a very simple model such as the constant function if Figure 2.4a will also result in poor results, as it is not able to capture the patterns at all. Model selection strikes a balance between those two extrema by taking the error of the model as well as its complexity into account.

Occam's Razor is a principle for model selection, saying one should pick the simplest model explaining the data reasonably well [Mur]. When fitting a model, the available data is usually scarce. Taking a simple model for small data sets makes sense, even if the data is generated from an arbitrarily complex model, since the available data might not suffice to reveal the true pattern and a complex model might start to model noise [Gru04]. Applying Occam's razor to the data set in Figure 2.4 would probably result in a polynomial of degree one, although the data is generated by a polynomial of degree three. There is just not enough data to say for sure.

Minimum Description Length is a formal version of Occam's Razor. In Minimum Description Length (MDL) learning is viewed as encoding, that is, a learning model encodes, i.e. compresses data by extracting respective patterns. The better the model is at finding the pattern, the shorter the code length. MDL mediates between model complexity and goodness of fit using the lengths of two different codes. The first code length is of the model that is used to describe the data. The more complex the model, the longer the code necessary to describe it. The data is then encoded with the model, yielding the second code length. The better the

[1] Random fluctuations being normally distributed with mean zero

model at describing the data, the shorter the code. MDL is then defined as the difference between the code length of the model, and the code length of the data encoded by the model. The model minimizing this difference is chosen [Roo16].

Model selection is very important for this thesis and used in Chapter 4 and Chapter 5. Formal definitions of MDL for the models used in this thesis are given in Section 2.4. Other information theoretic measures such as the Bayesian Information Criterion (BIC) or Akaike Information Criterion (AIC) exist. Those measures result in similar results as MDL [Pei14]. Other popular model selection techniques based on cross validation are difficult to apply due to the global dependencies in graph data [YJK+12]. MDL is derived in [Pei12, Pei13] for different types of graphs and variations of the model used in this thesis. Furthermore, MDL is successfully applied in [Pei14] for model selection, and thus chosen as means for model selection in this thesis.

2.3 Anomaly Detection

This section gives a short and general introduction into anomaly detection. Section 2.3.1 quickly sketches the difference between anomaly detection and related concepts. Section 2.3.2 introduces different types of anomalies, and Section 2.3.3 gives an overview over different techniques for anomaly detection. The terminology and concepts introduced in this section are important for the understanding of Chapter 5.

2.3.1 Anomaly Detection and Related Concepts

The goal of anomaly detection is to identify abnormal patterns in a data set. For this, a notion of *normal* must be established. Closely related to anomaly detection are noise removal and novelty detection. Anomaly detection distinguishes itself from noise removal by the *interestingness* of anomalies, that is anomalies carry relevant information [CBK09]. Consider the measuring of temperature on a local weather station and a tank in a chemistry factory. A sudden increase in the measured temperature of the local weather station can be discarded as an error in the sensor, and should be removed before summary statistics are reported. This is an example of noise, the abnormal measurement does not hold any relevant information. In contrast, an increase in temperature measured with the sensor in the factory cannot be discarded as sensor failure as easily. The increase could be an indication of something going wrong in the factory.

Novelty detection also aims at finding interesting patterns, deviating from the normal. Contrary to anomaly detection, the identified pattern are then incorpo-

rated into modeling the normal [CBK09]. An example is the social network of individuals that leave high school and enter college. They suddenly form relations to many people they have a low chance to even meet, let alone become friends with, given their previous social interactions. Instead of regarding this as an anomaly, it makes sense to incorporate this change in the prediction of future relation of those individuals.

The closeness of noise removal, novelty detection and anomaly detection makes the latter even more difficult. Additionally to identifying a deviation of the normal, differentiating between irrelevant, anomalous or a general change in normal behavior must be performed as well [CBK09].

2.3.2 Anomaly Types

Section 2.3.1 introduces an anomaly as deviation from the normal. Depending on the application domain, "deviation of the normal" might look quite different. Is a temperature reading of $100°C$ un-normal for a weather station in Munich? Most likely yes. What about $-5°C$? During summer it most likely is un-normal, while during winter it is not. This section differentiates between different types of anomalies to build a better understanding of the challenges inherent to anomaly detection, and provides a taxonomy for a better understanding later on.

Point Anomalies are individual data instances that can be regarded as un-normal with respect to the remainder of the data. Point anomalies are the most simple type of anomalies [CBK09]. For example the temperature reading of $100°C$ given earlier is an example of a point anomaly.

Contextual Anomalies are individual data instances that are un-normal in a specific context only. Consequently, each data point has two sets of attributes. One set is used to determine the *context* of the data point, and the other to specify the *behavior* of the data point. In order to identify an anomaly, the behavioral attributes of data instances with the same contextual attributes are compared. The temperature reading $-5°C$ given earlier is an example of a contextual anomaly [CBK09]. Here, the contextual attribute is the season and the behavioral attribute is the temperature.

Collective Anomalies are multiple related data instances that are un-normal with respect to the entire data set. Each data instance in the collective anomaly for itself is not necessarily an anomaly, only their co-occurrence marks them as un-normal [CBK09]. Using again the example of a weather station, the readings $28°C, 24°C, 29°C, 24°C, 26°C$ obtained in Munich over a period of one minute during

noon time on a summer day form a collective anomaly. Each reading alone is a perfectly sensible measurement, but together they can be regarded as un-normal.

Obviously, the type of anomaly that should be detected make the presence of certain attributes in the data necessary. While point anomalies may occur in any data set, contextual- and collective anomalies require the presence of a context or relation between data points respectively. Also, the point- and collective anomalies can be transformed into contextual anomalies by simply specifying a context [CBK09]. Likewise, contextual anomalies can be regarded as point- or collective anomalies *within* each context. Thus, techniques developed for one type of anomaly might also be used for others.

2.3.3 Anomaly Detection Techniques

This section introduces different techniques for anomaly detection. Although statistical techniques are used in this thesis, other techniques are introduced for the sake of completeness as well.

Classification Based techniques have two stages. A training phase during which a classifier is learned, and a testing phase, during which samples are classified as anomalous or normal. Two different approaches exist. The first one assumes the existence of multiple classes within the data set. A multi-class classifier is trained to discriminate between the different classes. In a one-vs-all setting, that is a binary classifier distinguishing between one class and all other classes, a data instance is classified as anomalous, if all one-vs-all classifier reject the data instance. Other multi-class classifiers can be used as well. Usually, such a classifier outputs a score for each class, that reflects how strongly the classifier beliefs a data instance belongs to this class. If the distribution is broad, that is the classifier is uncertain about the class, the data instance is deemed anomalous [CBK09].

Nearest Neighbor Based techniques assume normal data instances to form dense clusters and anomalous data instances to be (relatively) far away from other instances. Consequently, nearest neighbor based techniques require a function to evaluate the distance or similarity between two data points. A data instance is then labeled as normal or anomalous based on the distance/similarity to its k nearest neighbors [CBK09].

Clustering Based techniques are similar to Nearest Neighbor methods. In clustering, data instance are grouped together into clusters. There are three different flavors how clustering is used in anomaly detection. In the first one, normal data instances are assumed to belong to a cluster, while anomalous data instances do

not belong to any cluster. This assumption requires clustering techniques that do not force every data point to belong to a cluster [CBK09].

In the second one, normal data instances are assumed to be located close to their respective cluster centers, while anomalous data instances are located far away. Here a clustering of the data is obtained in a first step. In a second step the distance to the cluster centroid is calculated for each data instances is calculated and used to indicate anomalies [CBK09].

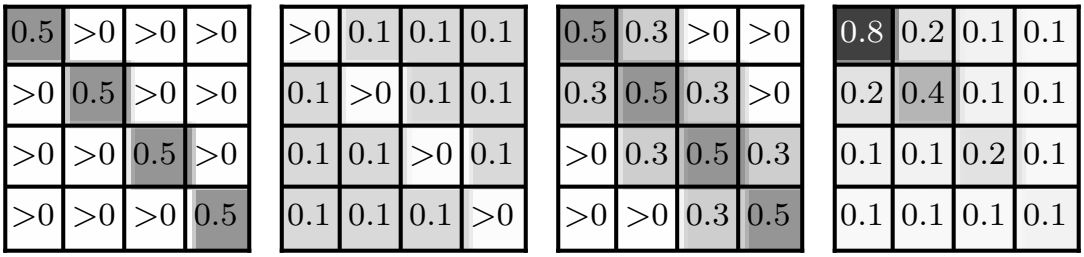
In the third one, normal data is assumed to form large and dense clusters, while anomalous data instances are assumed to either belong to small or sparse clusters. This techniques labels a whole cluster as anomalous if the size and/or density is blow a certain threshold [CBK09].

Clustering differs from nearest neighbor methods in the way, data instances are evaluated to identify anomalies. In nearest neighbor techniques, each data instances is evaluated with respect to their local neighborhood. In clustering each data instance is analyzed with respect to the cluster it belongs to [CBK09].

Statistical Techniques assume a stochastic model that generates the observed data. Normal data instances then occur in regions with high probability, while anomalies occur in regions with low probability. Statistical techniques are distinguished in parametric and non-parametric techniques. Parametric techniques can be used if the distribution of the data is known. Then, the parameters θ for the assumed distribution can be inferred from the data. For each data instance x, the anomaly score is then $p(x \mid \theta)^{-1}$, i.e. the inverse probability of the data point x under the assumed model given the inferred parameters. Alternatively, a hypothesis test can be used. The null hypothesis usually is that data instance x is generated from the assumed model parameterized with θ. If the null hypothesis is rejected, the data point x is deemed an anomaly. Hypothesis tests usually come with a test score, which can be used as anomaly score [CBK09].

Non-parametric techniques do not make assumptions about the distribution of the data, and instead is obtained from the data itself. A simple example of a non-parametric technique is the histogram. In a first step, a histogram is build from training data. In a second step, data instances are evaluated against the histogram. If they fall in any of the bins, they are deemed normal, else anomalous. Alternatively, the height of the bins can be used as anomaly score [CBK09].

Information Theoretic Techniques analyze the information content of a data set. The intuition behind information theoretic techniques is that anomalies induce irregularities in the information content of the data set. Consequently, the data set is more complex as the anomalies add information. Information theoretic



(a) assortative (b) dissortative (c) ordered (d) hierarchical

Figure 2.5: Block-matrices for $k = 4$ groups, representing different forms of large scale structure.

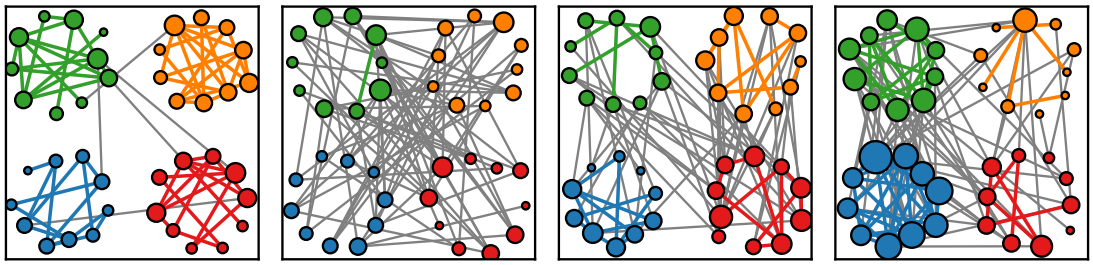
techniques thus work by identifying a subset of the data set, that maximizes the difference of the complexity of the original data set and the data set without the subset. The subset that maximizes this difference then contains the anomalies. Let $D \subset \Omega$ be a data set of any domain Ω and $f : \Omega \to \mathbb{R}$ a function that evaluates the complexity of a data set (for example entropy or Kolmogorov complexity). Then the set $\hat{S} = \text{argmax}_{S \subset D} \left(f(D) - f(D/S) \right)$ contains the anomalies [CBK09].

Spectral Anomaly Detection Techniques project the data into a low dimensional subspace in which normal and anomalous data instances can simply be differentiated. An example of such a technique is Principal Component Analysis (PCA). Data instances are labeled as normal or anomalous by considering their projection along the principal components with *low* variance. Normal instances will have a low deviation from the principal components, while anomalous instances will have a high deviation [CBK09].

All of the above techniques have advantages and disadvantages, strength and weaknesses. In order to choose a technique over the other it is paramount to have an understanding how anomalies relate to normal instances and of course what type of anomaly should be detected.

2.4 The Stochastic-Block-Model

This section introduces the probabilistic graphical models that form the heart of this thesis. Section 2.4.1 introduces the standard SBM for simple undirected graphs. Section 2.4.2 introduces a variant of the SBM, which generalizes to multigraphs. This simplifies mathematical treatment of the model. Also, the model introduced in Section 2.4.2 is used in the remainder of this thesis. Section 2.4.3 introduces a variant of the model presented in Section 2.4.2, specifically targeting



(a) assortative (b) dissortative (c) ordered (d) hierarchical

Figure 2.6: Graphs drawn from the distribution defined by the block matrices in Fig. 2.5. Each group contains ten vertices.

graphs containing groups with vertices having heterogeneous degrees. Section 2.4.2 and 2.4.3, give an Maximum Likelihood Estimate (MLE) for each respective parameter, and outline, how the best set of parameters can be chosen based on the MDL principle.

2.4.1 The Standard Stochastic Block Model

The Stochastic-Block-Model (SBM) is a probabilistic generative model for relational data, and has its origins in mathematical sociology [HLL83]. In social sciences, the SBM is used to identify communities of tightly interconnected vertices, but is not limited to this pattern. The SBM defines a parametric probability distribution $P(A \mid M, z)$ over graphs, where the parameters are:

- number of groups $k \in \mathbb{R}$,
- group membership of vertices $z \in \{1, \ldots, k\}^N$ where z_i gives the group of vertex i and N is the number of vertices,
- a Stochastic-Block-Matrix $M \in \mathbb{R}^{k \times k}$ where each entry M_{rs} gives the probability of a vertex in group r to connect to a vertex in group s.

Thus, a (complex) graph is reduced to a set of groups and inter- and intra group connectivity, with the central assumption: *vertices connect to other vertices solely based on their group membership*. Figure 2.7 shows the DGM for the SBM and illustrates this assumption. The dependence of edges on group membership is indicated by the arrows pointing from vertices z_i and z_j to observed vertex A_{ij}. The dependence on the respective entry in M is indicated by the arrow pointing from the vertex M_{rs} to vertex A_{ij}. The DGM in Figure 2.7 also illustrate the conditional independence of edges. As introduced in Section 2.2.2, a vertex in a DGM is conditionally independent from any other vertex in DGM given its parents.

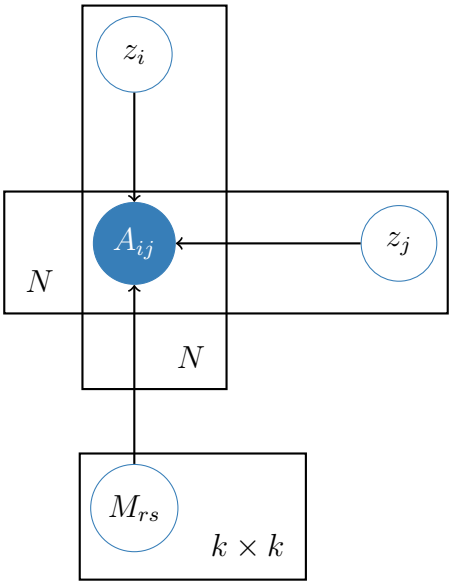


Figure 2.7: Graphical model for the SBM. vertex A_{ij} represents an element of an observed adjacency matrix. The remaining vertices z_i, z_j and M_{rs} are the unobserved parameters of the SBM.

Thus two edges A_{ij} and A_{lm} with $i \neq l$ and $j \neq m$ are conditionally independent given the respective entries in z and M, i.e., $A_{ij} \perp A_{lm} \mid z_i, z_j, z_l, z_m, M_{ij}, M_{lm}$, and follow a Bernoulli distribution ,i.e., $A_{ij} \sim \text{Ber}(M_{z_i z_j})$ and $A_{lm} \sim \text{Ber}(M_{z_l z_m})$. This property is important for the inference of the model parameters in subsequent sections.

From the DGM in Figure 2.7 it is easy to see that vertices of a graph share the same set of parameters. That is, vertices in one group share the same value in z and incident edges are governed by the same entries in M. Thus, vertices in one group are stochastically equivalent. This is very similar to the $G(n, p)$ model introduced previously in Section 2.1.3. In fact, the SBM can be regarded as a mixture of $G(n, p)$ models, where each pair of groups r, s is modeled with a different $G(n, p)$ model parameterized with the vertices belonging to the groups and $p = M_{rs}$. As a consequence, properties of vertices in a $G(n, p)$ model carry over to the SBM. As a matter of fact, the degree distribution of the SBM is a mixture of Poisson distributions. The SBM is thus able to generate graphs with a more realistic degree distribution compared to the $G(n, p)$ model, some of the limitations still apply, though.

As mentioned earlier, the SBM is used to identify groups of tightly interconnected vertices. This is also known as a-posteriori block modeling. That is, given an observed graph, find the parameters $\hat{M}, \hat{z}$, such that the likelihood of the SBM

parameterized with $\hat{M}, \hat{z}$ of generating the observed graph is maximized. Once the parameters are estimated, it is possible to generate, that is draw, graphs from the resulting distribution defined by the SBM (this is the topic of Chapter 5). How parameters are estimated in this thesis is a topic of the next section.

2.4.2 The Poisson Stochastic Block Model

This section outlines how parameters M and z are estimated given an observed adjacency matrix A. Furthermore, this section explains how to choose a value for parameter k, and how synthetic graphs are drawn from the resulting distribution. This is based on work in [KN11, Pei12, Pei13].

To simplify mathematical treatment, an undirected multi-graph is usually chosen as graph representation [KN11]. A directed version can be easily obtained by allowing M and A to be a-symmetric. The assumption of a multi-graph changes the model parameters given previously:

- adjacency matrix $A \in \mathbb{N}_0^{N \times N}$ with element A_{ij} for $i \neq j$ gives the number of edges between vertices i and j and twice the number of self-edges for $i = j$.
- Stochastic-Block-Matrix $M \in \mathbb{N}_0^{N \times N}$ where M_{rs} is the expected number of edges between groups r and s.
- Edges no longer follow a Bernoulli but a Poisson distribution: $A_{ij} \sim Poi(M_{z_i,z_j})$

Estimation of M By taking an MLE approach, M and z are chosen such that the probability $P(A \mid M, z)$, i.e., the a-posteriori likelihood, of generating exactly the observed edges is maximized. The probability is given by:

$$\begin{aligned} P(A \mid M, z) = &\prod_{i=1}^{N-1} \prod_{j=i+1}^{N} \frac{(M_{z_i z_j})^{A_{ij}}}{A_{ij}!} \exp(-M_{z_i z_j}) \\ &\times \prod_{i=1}^{N} \frac{(M_{z_i z_i})^{\frac{1}{2}A_{ii}}}{\frac{1}{2}A_{ii}!} \exp(-\frac{1}{2} M_{z_i z_i}). \end{aligned} \tag{2.19}$$

Equation (2.19) makes use of the conditional independence of edges mentioned earlier. Rearranging terms, dropping constants and taking the logarithm of (2.19) gives:

$$\log P(A \mid M, z) \propto \sum_{r=1}^{k} \sum_{s=1}^{k} \left(m_{rs} \log M_{rs} - n_r n_s M_{rs} \right), \tag{2.20}$$

where n_r is the number of vertices in group r and m_{rs} is the total number of edges running between groups r and s, or twice that much if $r = s$. The MLE of M_{rs} is then:

$$\hat{M}_{rs} = \frac{m_{rs}}{n_r n_s}, \tag{2.21}$$

where (2.21) is obtained by taking the gradient of the log-likelihood in (2.22) with respect to M_{rs}.

Estimation of z Taking the result from (2.21), plugging it into (2.20) and omitting constant terms, we obtain the unnormalized log likelihood:

$$\log P(A \mid z) \propto \sum_{r=1}^{k} \sum_{s=1}^{k} m_{rs} \log \frac{m_{rs}}{n_r n_s}. \tag{2.22}$$

Equation (2.22) depends solely on the group memberships z of the vertices. Additionally Equation (2.22) can be efficiently computed and gives rise to a heuristic algorithm to obtain $\hat{z}$ outlined in Figure **??** [KN11]. The algorithm works by firstly assigning a random group to each vertex. Then, for each vertex i Equation (2.22) is evaluated with respect to every group $1, \ldots, k$, while keeping the group of every other vertex fixed. The group that maximizes Equation (2.22) for vertex i is then taken as new group label for vertex i. At the end of the iteration over the vertices, the labeling yielding the largest value for Equation (2.22) is chosen as initialization for the next iteration. The algorithm terminates if no vertex changes labels during one iteration, i.e., if Equation (2.22) cannot be further increased. As the problem of estimating z is not convex, it is advised to run the algorithm in Figure 1 with different random initializations. The algorithm has a complexity of $\mathcal{O}[\mid \mathcal{V} \mid (k^2 + kD(G))]$ for a graph $G = (\mathcal{V}, \mathcal{E})$. When a vertex v changes from group r to group s, the respective counts for edges between groups r and s must be adapted. This takes time $k + d(v)$ and is done for all k groups. Thus, for all vertices, the vertex degree $d(\cdot)$ is replaced by the average degree $D(G)$. The complexity can be reduced by pre-computing the number of edges each vertex has to every group. These numbers only change, if one of the *adjacent* vertices changes its group. Assuming every vertex changes group on each iteration, except for the last, the upper bound for the complexity becomes $\mathcal{O}[\mid \mathcal{V} \mid (k + D(G)) + k \mid \mathcal{V} \mid]$.

Graph Generation Given a choice for M and z, a graph is generated from the SBM, by considering each possible edge $(v, w) \in \mathcal{V} \times \mathcal{V}$ and drawing the number of edges between v and w from a Poisson distribution with:

$$(v, w) \sim \begin{cases} \text{Poi}(\frac{1}{2} M_{rs}) & \text{if } r = s \text{ and the graph is undirected} \\ \text{Poi}(M_{rs}) & \text{else} \end{cases} \tag{2.23}$$

Algorithm 1 Algorithm based on label switching to estimate parameter z of the SBM, given a choice of parameter M

$z := random\_initialization()$
$\max L := 0$
while not converged **do**
  for $i = 1$ to N **do**
    $\hat{l} := 0$
    $\hat{z}_i := z_i$
    for $r = 1$ to K **do**
      $z_i :=$ r
      $l := likelihood()$
      if $(l > \hat{l})$ or $(r = 1)$ **then**
        $\hat{l} :=$ l
        $\hat{z}_i := z_i$
      end if
    end for
    $z_i := \hat{z}_i$
    if $(\hat{l} < \max L)$ or $(i = 1)$ **then**
      $\hat{z} = z_i$
    end if
  end for
end while

Model Selection A remaining problem is how to choose the free parameter k. The un-normalized log likelihood in Equation (2.22) cannot be used to determine k, as the likelihood increases with increasing k. Thus, a good choice for k must allow the model to capture the essential structure, while keeping it from fitting to noise in the data. This is analoguous to the example given in Section 2.2. MDL introduced in Section 2.2.4 provides a possible solution to this problem. The MDL for a SBM defined over a un-directed multi-graph $G = (\mathcal{V}, \mathcal{E})$ is derived in [Pei12] and [Pei13]. The entropy of a SBM is defined as [Pei12]:

$$H(m) := \frac{1}{2}\sum_{r=1}^{k}\sum_{s=1}^{k}(n_r n_s + e_{rs})h\left(\frac{n_r n_s}{n_r n_s + e_{rs}}\right), \tag{2.24}$$

where h is the binary entropy function:

$$h(x) := -x\log(x) - (1-x)\log(1-x). \tag{2.25}$$

The information necessary to describe the model is [Pei13]:

$$I(m) :=\mid \mathcal{E} \mid h\left(\frac{k(k+1)}{2 \mid \mathcal{E} \mid}\right) + \mid \mathcal{V} \mid \log(k), \tag{2.26}$$

where h again is the binary entropy function as defined in Equation (2.25). Thus the MDL becomes:

$$MDL(m) := I(m) - H(m). \tag{2.27}$$

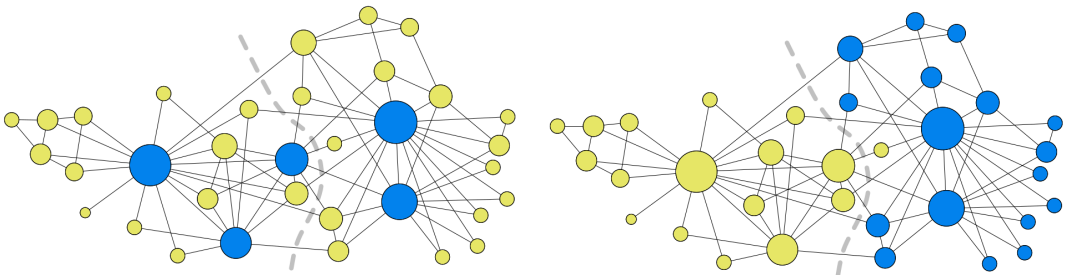
In case of a directed multi-graph, only the entropy changes to:

$$H(m) := \sum_{r=1}^{k}\sum_{s=1}^{k}(n_r n_s + e_{rs})h\left(\frac{n_r n_s}{n_r n_s + e_{rs}}\right), \tag{2.28}$$

i.e. only the factor $1/2$ is dropped [Pei12]. Parameter k is then given by the value that minimizes the MDL of the SBM. As this requires the estimation of parameters M and z for each value of k, this approach is computationally expensive.

2.4.3 The Degree Corrected Stochastic Block Model

One of the "issues" with the SBM is its tendency to group vertices according to their degree. The authors in [KN11] propose a degree-corrected variant of the SBM, the DCBM, by introducing a new parameter θ_i for each vertex, modeling the expected degree. Figure 2.8 illustrates the different group separations found by SBM and DCBM. Figure 2.8a shows the separation found by the SBM and DCBM for a graph of social relationships in a karate club. The club is separated



(a) Separation with SBM (b) Separation with DCBM

Figure 2.8: Separation for Zachary's Karate Club found by SBM and DCBM. The SBM separates vertices by degree, the DCBM correctly infers the underlying structure. Figure taken from [KN11].

into two fractions indicated by the dashed line. As Figure 2.8a shows, the SBM fails to detect the underlying group structure. In contrast, the DCBM correctly identifies the underlying group structure, as Figure 2.8b illustrates.

This thesis is concerned with graphs arising in the context of communication networks. The degree distribution of such graphs is known to follow a power-law [New10, AGL15, AB02]. Therefore, vertices in such graphs might express structure, that cannot be captured by the SBM. This thesis thus also evaluates graphs with respect to the DCBM.

The remainder of this section gives a MLE for each of the parameters for the directed and undirected version of the DCBM. To derive the estimates, undirected and directed multi-graphs are assumed respectively.

Inference of parameters $\theta, \mathbf{M}$ - undirected The goal is to find values for M and θ that maximize the likelihood. The probability of an undirected multi-graph, given by adjacency matrix A is:

$$\begin{aligned} P(A \mid M, z, \theta) = & \prod_{i=1}^{N-1} \prod_{j=i+1}^{N} \frac{(\theta_i \theta_j M_{z_i z_j})^{A_{ij}}}{A_{ij}!} \exp(\theta_i \theta_j M_{z_i z_j}) \\ & \times \prod_{i=1}^{N} \frac{(\theta_i^2 M_{z_i z_i})^{\frac{1}{2} A_{ii}}}{(\frac{1}{2} A_{ii})!} \exp(\theta_i^2 M_{z_i z_i}). \end{aligned} \tag{2.29}$$

As before, Equation (2.29) makes use of the conditional independence of edges. Rearranging terms, dropping constant expressions and taking the logarithm yields:

$$\begin{aligned}\log P(A \mid M, z, \theta) \propto & 2\sum_{i=1}^{N} d(v_i)\log\theta_i \\ & + \sum_{r=1}^{k}\sum_{s=1}^{k}(m_{rs}\log M_{rs} - M_{rs}),\end{aligned} \tag{2.30}$$

from which the MLE estimates for θ and M are derived by setting the gradient of Eq. (2.30) equal to zero:

$$\hat{\theta}_i = \frac{d(v_i)}{\kappa_{z_i}}, \quad \hat{M}_{rs} = m_{rs}, \tag{2.31}$$

where m_{rs} is the total number of edges running between communities r and s, and κ_{z_i} is the sum of degrees in group z_i.

Inference of parameter z - undirected Similar to Section 2.4.2 the unnormalized log-likelihood can be written in sole dependence of z, by plugging the respective MLE solutions into Eq (2.30):

$$\log P(A \mid z) \propto \sum_{r=1}^{k}\sum_{s=1}^{k} m_{rs}\log\frac{m_{rs}}{\kappa_r\kappa_s}. \tag{2.32}$$

The same heuristic algorithm used for the SBM can be used to obtain the group membership of the vertices. Equation (2.22) is replaced as objective function by Equation (2.32).

Inference of parameters $\theta, \mathbf{M}$ - directed The DCBM cannot be applied to directed graphs as readily as the SBM. Since vertices in directed graphs have in general different in- and out-degrees, the directed version of the DCBM, the Directed Degree Corrected Block Model (DCBM), needs two parameters per vertex. One parameter to model the expected in- and the other to model the expected out-degree [ZYM12]. The probability of an observed directed multi-graph is then:

$$\begin{aligned}P(A \mid M, z, \theta) = & \prod_{i=1}^{N}\prod_{j=1}^{N}\frac{(\theta_i^{out}\theta_j^{in}M_{z_iz_j})^{A_{ij}}}{A_{ij}!}\exp(\theta_i^{out}\theta_j^{in}M_{z_iz_j}) \\ & \times\prod_{i=1}^{N}\frac{(\theta_i^2 M_{z_iz_i})^{\frac{1}{2}A_{ii}}}{(\frac{1}{2}A_{ii})!}\exp(\theta_i^{out}\theta_i^{in}M_{z_iz_i}).\end{aligned} \tag{2.33}$$

As before, Equation (2.33) makes use of the conditional independence of edges. Rearranging terms, dropping constant expressions and taking the logarithm yields:

$$\begin{aligned}\log P(A \mid M, z, \theta) \propto \sum_{i=1}^{N} \left(d^{in}(v_i) \log \theta_i^{in} + d^{out}(v_i) \log \theta_i^{out}\right) \\ + \sum_{r=1}^{k}\sum_{s=1}^{k}(m_{rs} \log M_{rs} - M_{rs}),\end{aligned} \tag{2.34}$$

where κ_r^{out} is the sum of out-degrees, and κ_r^{in} the sum of in-degrees of group r. Taking the gradient with respect to Equation (2.34) yields the estimates:

$$\hat{\theta}_i^{in} = \frac{d^{in}(v_i)}{\kappa^{in}}, \quad \hat{\theta}_i^{out} = \frac{d^{out}(v_i)}{\kappa^{out}}, \quad \hat{\omega}_{rs} = m_{rs} \tag{2.35}$$

Inference of parameter z - directed As before, the MLE of the parameters can be used to write the log-likelihood in sole dependence of z:

$$\log P(A \mid z) \propto \sum_{r=1}^{k}\sum_{s=1}^{k} m_{rs} \log \frac{m_{rs}}{\kappa_r^{in}\kappa_s^{out}}. \tag{2.36}$$

Equation (2.36) can then be used as objective function for the label switching algorithm presented in Section 2.4.2.

Graph Generation Given a choice for M and z, a graph is generated from the DCBM, by considering each possible edge $(v, w) \in \mathcal{V} \times \mathcal{V}$ and drawing the number of edges between v and w from a Poisson distribution with $\lambda = \theta_i\theta_j M_{z_v z_w}$, i.e.:

$$(v, w) \sim \begin{cases} \text{Poi}(\theta_i\theta_j M_{z_v z_w}) & \text{if } v \neq w \\ \text{Poi}(\frac{1}{2}\theta_i\theta_j M_{z_v z_w}) & \text{else.} \end{cases} \tag{2.37}$$

In the case of the DDCBM, edges are drawn according to:

$$(v, w) \sim \text{Poi}(\theta_i^{out}\theta_j^{in} M_{z_v z_w}) \tag{2.38}$$

Model Selection - undirected Similar to the SBM, MDL can be used to decide between different sets of parameters. In case of the DCBM, the entropy is:

$$H(m) := -\mid \mathcal{E} \mid -\sum_{d \in \mathcal{D}} \mid \mathcal{V}_d \mid \log(d!) - \frac{1}{2}\sum_{r=1}^{k}\sum_{s=1}^{k} \log\left(\frac{\mid \mathcal{E}_{rs} \mid}{\mid \mathcal{V}_r \mid\mid \mathcal{V}_s \mid}\right), \tag{2.39}$$

where $\mathcal{D}$ is the set of all degrees in G, $\mathcal{V}_d := \{v \in \mathcal{V} \mid d(v) = d\}$ is the set of all vertices with degree d, $\mathcal{V}_r := \{v \in \mathcal{V} \mid z_v = r\}$ is the set of all vertices in group r, and $\mathcal{E}_{rs} := \{(v, w) \in \mathcal{E} \mid z_v = r \wedge z_w = s\}$ is the set of all edges running between groups r and s. The information necessary to describe the model is [Pei13]:

$$I(m) := - \mid \mathcal{E} \mid h\left(\frac{k(k+1)}{2 \mid \mathcal{E} \mid}\right) + \mid \mathcal{V} \mid \log(k) - \mid \mathcal{V} \mid \sum_{d \in \mathcal{D}} p_d \log p_d, \tag{2.40}$$

where p_d is the probability of degree d and h the binary entropy function defined in Equation (2.25). The first part of Equation (2.40) is just Equation (2.26). As before, the MDL is then simply:

$$MDL(m) := H(m) - I(m). \tag{2.41}$$

Model Selection - directed The entropy for the DDCBM is [Pei12]:

$$\begin{aligned} H(m) := &- \mid \mathcal{E} \mid - \sum_{d \in \mathcal{D}^{in}} \mid \mathcal{V}_d \mid \log(d!) - \sum_{d \in \mathcal{D}^{out}} \mid \mathcal{V}_d \mid \log(d!) \\ &- \frac{1}{2} \sum_{r=1}^{k} \sum_{s=1} k \log\left(\frac{\mid \mathcal{E}_{rs} \mid}{\mid \mathcal{V}_r \mid\mid \mathcal{V}_s \mid}\right), \end{aligned} \tag{2.42}$$

where $\mathcal{D}^{in}$ and $\mathcal{D}^{out}$ are the sets of all in- and out degrees of graph G respectively. And the information necessary to describe the model is [Pei13]:

$$\begin{aligned} I(m) :=&- \mid \mathcal{E} \mid h\left(\frac{k^2}{\mid \mathcal{E} \mid}\right) + \mid \mathcal{V} \mid \log(k) \\ &- \mid \mathcal{V} \mid \left(\sum_{d \in \mathcal{D}^{in}} p_d \log p_d + \sum_{d \in \mathcal{D}^{out}} p_d \log p_d\right), \end{aligned} \tag{2.43}$$

The MDL for the DDCBM is then again simply the difference between entropy and information.

Chapter 3

Related Work

This chapter introduces work related to the topics discussed in the remainder of this thesis. Chapter 4 discusses the identification of meaningful groups in communication networks, Chapter 5 synthetic graph generation and Chapter 6 anomaly detection. This chapter is accordingly organized in three sections. Section 3.1 presents existing work with respect to identification of groups in communication networks. Section 3.2 introduces existing models and generators for synthetic graphs, and Section 3.3 briefly introduces related work in anomaly detection similar to the approach discussed in this thesis.

3.1 Community Detection in Communication Networks

Community detection in communication networks has been investigated in the context of traffic classification and network monitoring. The underlying idea is that hosts consuming the same services have similar connection patterns. By grouping them together, few hosts give away the used service(s) in the community, enabling traffic classification and content management in the presence of encryption and obfuscation [Ili09, IGER+10]. The found communities can also be leveraged as profiles to detect possible malicious users or applications [XWG11, JGS+16]. For example [IGER+10] uses the IP-to-IP communication graph together with a heuristic community detection algorithm based on modularity for traffic classification. Another approach to detect communities of similar hosts in [XWG11] is based on modeling interactions of end hosts as bipartite graphs. A one-mode projection of the bipartite graphs is then derived to obtain a sort of similarity graph. Spectral clustering is applied to obtain communities. Authors in [JGS+16] follow the same approach, but use modularity instead of spectral clustering for the detection of communities. Authors in [JGL15] and [DHB+10] detect groups of similar hosts

by by projecting them into a feature space, and cluster them there. Considered features are for example the distribution of source and destination ports, mean packet size, average duration of flows and many more. Work in [AKM+05] infers so called communities of interest (COIs) for each host in an enterprise campus network. The COI of a host (henceforth called target host) is very broadly defined as all hosts the target host interacts with. The authors then provide more restrictive definitions tailored to specific use cases such as resource allocation or intrusion detection. Work in [TPGK03] is most similar to the methodology in this thesis. The authors aim to group hosts in an enterprise network according to their role (developer, sales clerk, etc.) solely based on their connection pattern. The authors use a simple heuristic approach, similar in spirit to clustering, that maximizes the similarity of hosts within one group, and the dissimilarity between groups. Similarity between two host is defined as the number of neighbors the two hosts share.

The approach taken in this thesis differs from previous work in that probabilistic graphical model are used to infer groups of vertices. Authors in [Ili09, IGER+10, XWG11, JGS+16] use the general definition of communities, i.e., high within group density and low between group density. The probabilistic models considered in this thesis go beyond simple community detection and identify groups of vertices with similar roles. In that regard, the goal in this thesis is comparable to that of [AKM+05] and [TPGK03]. What sets the approach of this thesis apart, is the use of principled methods grounded in probability and information theory for identification of roles. This allows the quantification of how well the inferred probabilistic model explains the data, as well as the application of model selection techniques for choosing the best among competing models. Contrary to work in [DHB+10] and [JGL15] the approach in this thesis does not project vertices into some space, but directly uses the graph, i.e., the relational data.

3.2 Generation of Synthetic Graphs

In the area of communication networks, popular models for generating synthetic graphs already exist [Wax88, New10]. For instance, in the Waxman model [Wax88], the probability of an edge is a function of the spatial location of vertices. In the preferential attachment model, the probability of an edge between two vertices depends on the vertex degree. The Watts-Strogatz model generates networks with the small world property [New10].

Many network generators exist that build upon the three mentioned models. For example, GT-ITM [CDZ97] is based on the Waxman model and generates graphs by separately modeling hierarchical levels of the Internet graph. The more recent generator GeoTopo [HZRR15] is similar in principle to GT-ITM and in-

corporates not only hierarchy, but also considers geographic, demographic and economic information for topology generation. BRITE [MLMB01] is based on preferential attachment and targets the connectivity pattern of the Internet graph by generating a degree distribution following a power-law. INET [WJ] is similar to BRITE and also combines structural and degree information. S-BITE [AGL15] separates the Internet graph into different levels, models the connectivity between and within the levels and uses specific degree distributions, as well as preferential attachment and small world mechanisms. In contrast, Orbis [MHK+07] generates graphs according to a given (joint) degree distribution. Furthermore, Orbis is able to generate graphs of different sizes by scaling the distribution and is applicable to arbitrary graphs. Orbis is used for comparison in Chapter 5.

The approach in this thesis differs in that structural properties of a graph are learned in an unsupervised fashion by estimating the parameters of a model representing a probability distribution over graphs. Extensive analysis of empirical network data, which was necessary for the design of existing network generators, is replaced by inferring model parameters on that data, for which principled methods exist. The probabilistic models can be used as a black box, which takes the modeling effort from the user. The probabilistic models are applicable to arbitrary graphs due to their generality. i.e., they do not impose a certain structure a-priori. This is in contrast to most of the existing generators [MLMB01, AGL15, CDZ97, HZRR15, WJ]. Synthetic graphs can be generated by sampling from the inferred distribution.

3.3 Network Based Malware Detection

Using communication patterns of hosts to identify anomalies, e.g., malware has been investigated previously. Detection of worms in [EAAT04] relies on three signatures expressing the behavior of single hosts and the behavior of a group of hosts. The signatures are specifically designed to identify infected hosts. Authors in [SBN12] model the behavior of worms during infection phase. That is, they create a signature of network events, which can be used to distinguish normal and anomalous traffic. A similar approach is taken in [HSCZ10]. Aside from host to host communications, authors in [HSCZ10] additionally integrate expert knowledge regarding the behavior of applications and services. Approaches summarized in [Sta12] rely on the number of failed connections per host, or the increase in the number of novel neighbors for a host between two neighboring time steps. A more elaborate technique described in [Sta12] extracts protocol-specific graphs and monitors statistics for graph properties of those graphs in order to detect malware. Authors in [MABXS10] analyze a large set of malware samples and create behavioral profiles for hosts by mapping them into a numeric feature space. Clustering

is then applied to identify anomalous hosts. Similar approaches in spirit to those presented so far are taken in most of the works described in [FSR09] and [GZC14].

The approach for malware detection, i.e., identification of hosts in a campus network being part of a botnet, in this thesis differs from previous work in several aspects. Detection is based solely on the IP-to-IP communication between hosts. The graph is separated into groups of similar hosts using a probabilistic generative model. The probability of observed connections for each host under the probabilistic model is then used to detect deviations from behavior defined and labeled as normal in respective groups. This approach is based only on relational data between host, that is, only on the graph extracted from the IP-to-IP communication. Thus no additional features, such as median flow duration, median package size, port entropy or others, are used.

Chapter 4

Service Graph

This chapter interprets the separation found by the DCBM with respect to the services found in an IP-to-IP communication graph. Specifically, this chapter answers the following questions:

1. Consume or provide hosts in one group specific services or are they evenly distributed?
2. Are servers and clients assigned to the same or different groups?
3. Do groups correspond to IP-subnets?
4. Is the size of packages sent between groups evenly distributed?

These answers will be discussed once for an overlay-graph based on the IP- and port information of hosts, and for a graph based only on IP-Addresses. Depending on the answers of the above questions, the structure inferred by the DCBM can be used for different applications. If specific groups consume certain services only, characteristics of those services can be used for, e.g., routing decisions, content management or resource allocation.

The remainder of this chapter firstly introduces the used graphs and how they are obtained and then discusses the results giving answers to the above questions.

4.1 Materials and Methods

This section introduces the techniques used to extract two different graphs from a packet capture (PCAP) file, as well as metrics and methodologies to assess whether inferred groups are meaningful. The structure is as follows, Section 4.1.1 describes the extraction of graphs from a PCAP file. Section 4.1.2 describes various measures to quantify properties of hosts in different groups. Those measures are used in

Graph	Order	Edges	Density	Coreness	Neighbor Degree	Clustering Coefficient
Port-Graph	3 152	13 932	$1.40e^3$	6.34	122.66	0.0561
IP-Graph	2 581	12 329	$1.83e^3$	6.72	155.0	0.1012

Table 4.1: Graph metrics for Service- and IP-Graph

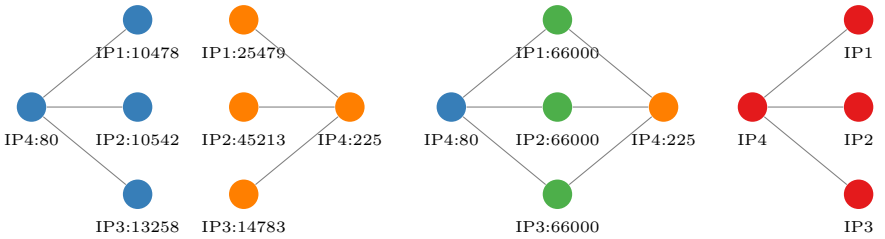
IP Graph		Port-Graph	
Service	Occurrence	Serivce	Occurrence
Client	1 856	Client	1 534
IP-LP	786	http	773
http	761	ntp	240
ntp	244	netbios-ns	190
netbios-dgm	243	svrloc	110
netbios-ns	212	https	76
svrloc	113	ipcserver	76
ipcserver	97	524	26
dhcp	82	snmp	14
https	74	ssh	14
Remainder	257	Remainder	161

Table 4.2: Top ten most frequent Services. *Client* refers to labels with a port larger 1 024, IP-LP refers to vertices associated with any IP layer protocol.

Section 4.2 and Section 4.3 to assess whether inferred group express distinctive differences regarding consumed services, topological role, contained IP-subnets and sent as well as received traffic.

4.1.1 Graph Extraction and Representation

To investigate whether probabilistic graphical models, namely the Directed Stochastic Block Model (DSBM) and Directed Degree Corrected Block Model (DCBM), can be used to infer meaningful groups in IP networks, two different graphs extracted from the same PCAP file are considered. Both graphs are extracted from a PCAP file from the Lawrence Berkeley National Laboratory (LBNL). The capture of port 003 recorded at 15.12.2004 and is publicly available at [ber]. The first graph, which is considered in this chapter, is based on port and IP information, and referred to as Port-Graph for the remainder of this chapter. The other graph is based solely based on IP addresses and henceforth



(a) Full-Graph (b) Port-Graph (c) IP-Graph

Figure 4.1: Difference between Full-, Port-, and IP-Graph. The part in front of the colon in each label symbolizes an IP-address. The part after the colon represents some port.

referred to as IP-Graph. The following paragraphs detail the extraction process.

Port-Graph Each vertex in the graph represents an IP and port pair encountered in the trace. A graph could be extracted from the capture, where each vertex is a distinctive IP-Port pair. A graph constructed in this way is referred to as Full-Graph going forward. The Port-Graph is constructed in this fashion. However, ports larger than 1 024 are set to the value 66 000, i.e., a value outside of the possible range of ports. IP source and destination pairs without TCP or UDP port information are assigned a value of -1 as port. Tis are IP layer protocols such as ICMP. Figure 4.1 illustrates why high value ports are set to the same value. Figure 4.1 shows a Full-, Port-, and IP-Graph based on the same IPs and ports. Figure 4.1a shows a Full-Graph with vertices consuming/providing two services. As Figure 4.1a shows, the graph is unconnected. A Full-Graph will in general consist of many disconnected components corresponding to the different services, and thus has little structure. This representation is thus not suited for the purpose of this thesis. By masking ports larger 1 024, unconnected components are merged. This is illustrated in Figure 4.1b. The separate client hosts for each service from Figure 4.1a are now one vertex respectively. It is now possible to identify server and client groups, where clients share the same services. Hosts with a port larger 1 024 are referred to as clients, and hosts with a port smaller 1 024 as servers in the remainder of this thesis.

Table 4.2 lists the top ten most frequent services for each graph, i.e., the column *Occurrence* gives the number of vertices that are associated with the respective service. As Table 4.2 shows, there are, even among the top ten, few very popular services accounting for most of the vertices. The most frequent service is 66 000, i.e., the placeholder for clients, which is associated with 1 534 vertices, or roughly half the number of vertices in the Port-Graph. Hypertext Transfer Pro-

tocol (HTTP) accounts with 773 vertices for a quarter of the vertices, followed by NTP with 240 vertices. Simple Network Management Protocol (SNMP) and Secure Shell Protocol (SSH) are the tail of the list with 14 associated vertices each. Services not explicitly listed in Table 4.2 are aggregated in the label *Remainder*, which accounts for 161 vertices. According to these numbers, about 50 % of the vertices in the graph provide services, and the other 50 % do only consume services.

IP Graph Each vertex in this graph corresponds to a single IP-address encountered in the trace. Thus there is no longer a sharp distinction between vertices offering services and consuming services. One vertex in the graph is now associated with sometimes hundreds of distinctive ports. Figure 4.1c illustrates this process. The hosts providing different services in Figure 4.1b are now one vertex in Figure 4.1c.

Table 4.2 lists the frequency of each service in the IP-Graph. In case of the IP-Graph, the table *Occurrence* does not sum to the number of vertices in the graph, as each vertex might be associated with multiple services. As for the Port-Graph, the client placeholder 66 000 is associated with most of the vertices, i.e., with 1 856 out of 2 581 vertices. The placeholder -1 follows with 786 vertices, closely followed by HTTP with 761 vertices. Hypertext Transfer Protocol Secure (HTTPS) accounts with 74 vertices for the smallest fraction. Services not listed in the table are aggregated in the label *Remainder* and are associated with 257 vertices. Summing the occurrences of services in the IP-Graph results in a larger value than for the Port-Graph. Hosts that were not part of the largest connected for the Port-Graph are contained in the IP-Graph due to the collapsing effect illustrated in Figure 4.1.

Table 4.1 gives graph features of the both graphs. The IP-Graph has 571 vertices and 1 603 edges less than the Port-Graph. Thus the majority of the vertices of the Port-Graph consists of an IP-Address with only one port associated, and consequently, the IP-Graph contains these vertices as well. The density of the IP-Graph is higher than the density of the Port-Graph. The difference of $4e^{-4}$ does not seem much. However, the IP-Graph would need only 9 323 instead of the 13 329 edges to preserve the density of the Port-Graph. This indicates that clients connect to servers with varying IP addresses, as only servers are collapsed in the step from the Port- to the IP-Graph. More precisely, there are dedicated machines for different services. If one machine would host a large number of popular services, the number of the edges in the IP-Graph would be reduced stronger.

Both graphs are investigated in order to establish whether the IP information alone suffices to find meaningful groups. If the IP-Graph yields similar results than the Port-Graph, then this would provide several advantages. The IP-Graph

is generally smaller, resulting in less computation needed to infer groups. Also, the Port-Graph might suffer from obfuscating, that is applications mask their ports [IGER+10]. Thus, hosts that actually are servers could be cast to clients.

4.1.2 Community Quality Measure

This section covers measures and methods to assess how reasonable the inferred groups are.

Enrichment is a measure to assess how similar the vertices in one group are, and is based on additional meta data providing additional information about the vertices, which is not used during inference of the groups. Vertices that are similar are assumed to have more meta-data in common. To quantify this, authors in [ABL10] propose the computation of enrichment for groups between pairs of vertices:

$$E(r) := \frac{\mid \mathcal{V} \mid (\mid \mathcal{V} \mid -1) \sum_{(v,w)\in g(r)\times g(r)} s(v,w)}{\mid g(r) \mid (\mid g(r)-1 \mid) \sum_{(v,w)\in \mathcal{V}\times\mathcal{V}} s(v,w)}. \tag{4.1}$$

The function $g : \mathbb{N} \to \mathcal{V}$ gives for a group index r the set of vertices belonging to this group. The function $s : \mathcal{V} \times \mathcal{V} \to \mathbb{R}^+$ is a similarity measure defined over the meta-data of two vertices. In essence, Equation 4.1 divides the average similarity of one group by the overall similarity of all vertices in a graph $G = (\mathcal{V}, \mathcal{E})$. The denominator serves as a baseline similarity and large values for the enrichment indicate a high similarity of the vertices in a group[1]. A simple indicator function is used as similarity measure for the Port-Graph:

$$s(v,w) := \begin{cases} 1, & \text{if port of } v \text{ equals port of } w, \\ 0, & \text{else.} \end{cases} \tag{4.2}$$

In the IP-Graph, one vertex has multiple associated services, i.e., ports. Therefore the Jaccard-Similarity is used as similarity measure:

$$d(v,w) := \frac{service(v) \cap service(w)}{service(v) \cup service(w)}, \tag{4.3}$$

where $service(\cdot)$ gives the services associated with a vertex. Both measures yield values in the interval $[0, 1]$.

The advantage of enrichment over measures such as the entropy is the inclusion of the background enrichment. The following three cases exist:

[1] A similarity measure is not a distance measure. For a distance measure $d(x,y) = 0 \leftrightarrow x = y$, however for a similarity measure $d(x,y) < d(x,x)$ for $x \neq$. That is the similarity measure is maximal for equal values [Run].

1. $E \sim 1$: Distribution of meta-data within group equals the overall distribution, i.e. the group is a random sample with respect to the meta-data. Consequently, vertices are not grouped according to services they consume or provide.

2. $E < 1$: Vertices within group have less meta-data in common compared to all pairs, i.e. vertices with different services are grouped together.

3. $E > 1$: Vertices within group have more meta-data in common compared to all pairs, i.e. vertices within group consume/provide the same service.

The cases correspond to the questions at the beginning of this chapter. Case two answers the question, whether servers and clients are assigned to the same group. As vertices consuming services all share the value 66 000, for the group to be heterogeneous, vertices with different ports smaller 1024 must be in that group. Thus, many different servers are in the group. Similarly, if many groups have $E > 1$, vertices in those groups share the same services. Taking simply the entropy of a group with respect to the services of the contained vertices does not allow such a direct interpretation.

Identification of Server and Host groups is done by investigating the ports a group sends to or receives traffic from:

- source ports of traffic originating within the group (SRC-TO),
- destination ports of traffic originating within the group (DST-TO),
- source ports of traffic that is send to the group (SRC-FROM),
- destination ports of traffic that is send to the group (DST-FROM).

SRC-TO and DST-FROM are ports of vertices within the group, DST-TO and SRC-FROM are ports of vertices not in the group. Basically, occurrences of source and destination port of interaction originating within the group or terminating in the group are counted. Then the entropy over the ports is taken for each group. Taking the entropy takes into account the frequency with which a port occurs, which simply noting distinctive ports does not. If, for example, two different ports occur in SRC-IN, then it is a difference if one port accounts for 99 % and the other for only 1 %, compared to the case, in which both account for 50 %.

This approach yields four scalar values for each group, which allow to distinguish between groups containing server and groups containing hosts. Table 4.3 gives possible patterns of the four values and their interpretation.

Label	DST-FROM	SRC-FROM	DST-TO	SRC-TO	Interpretation
Host	high	low	low	high	Group receives on and sends from many different ports to a small number of ports. Thus the group contains hosts.
Server	low	high	high	low	Group receives on and sends from a small number of ports to many different ports. Thus the group contains server.
Homo-gene-ous	low	low	low	low	Group receives on and sends from a small number of ports while receiving from and sending to only a small number of ports. Thus the group and its adjacent groups offer and consume a small number of services.
Mixed	high	high	high	high	Group receives on and sends from a large number of ports, while receiving from and sending to a large number of ports. Thus the group and its adjacent groups either contain a mix of server and hosts or it is Peer-to-Peer (P2P) Traffic.

Table 4.3: Pattern of the entropy values of different port groups obtained by considering the bidirectional communication between groups. Other patterns also occur, but lack a clear interpretation and are thus labeled as unknown.

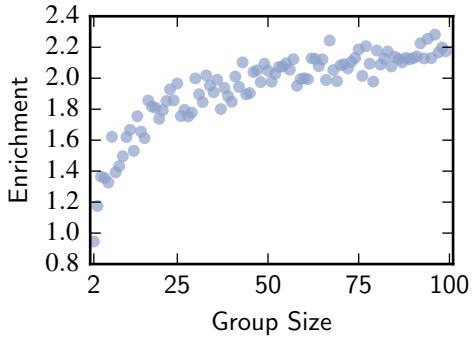


Figure 4.2: Average Enrichment over all DDCBM Models on Port-Graph

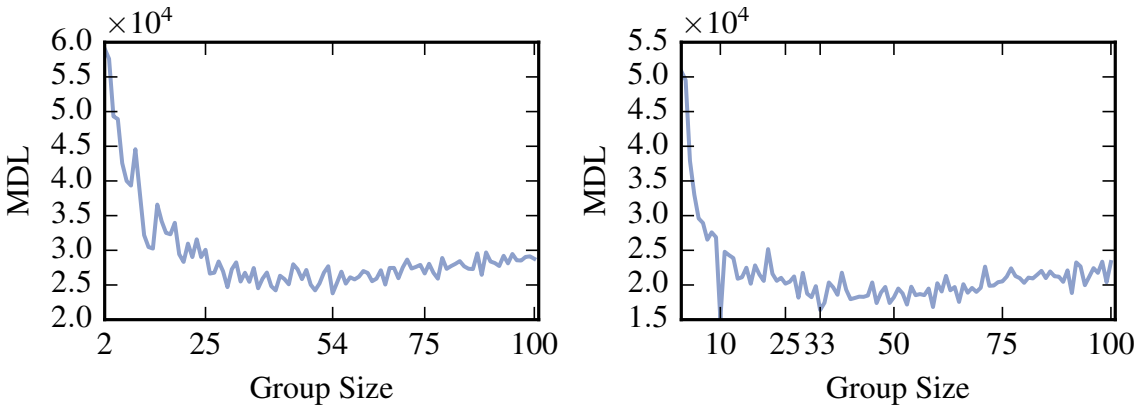
Model selection is necessary, as the exact number of groups is not known. The number of services could serve as an indicator, however, hosts of different services could be described with a single group or hosts using one service could form a complex pattern yielding multiple groups. Thus, models with $k = 2, \ldots, 100$ are fitted and MDL is used to select the optimal k. Enrichment cannot be used for this task as it would lead to overfitting. Section 4.2.1 and Section 4.3.1 illustrate this.

4.2 Evaluating Groups inferred with the DD-CBM

This section discusses the results obtained for the Port-, and IP-Graph obtained with the DDCBM. Section 4.2.1 outlines the effect of parameter k on enrichment and motivates model selection, which is discussed in Section 4.2.2. Section 4.2.3 discusses the results obtained with the DDCBM for enrichment. Section 4.2.4 uses port entropies to distinguish between server and client groups. Section 4.2.5 investigates whether groups correspond to IP subnets, and Section 4.2.6 analyzes sent and received traffic.

4.2.1 Influence of Parameter k

Figure 4.2 shows a scatter plot with the average enrichment on the y-Axis and the group size on the x-Axis for DDCBM models with parameter $k = 1, \ldots, 100$ estimated on the Port-Graph. Figure 4.2 highlights the need for a principled method for model selection, as enrichment increases with the complexity of the model and is thus not suited as a criterion to decide which model to use. Figure 4.2 illustrates this trend. This process is analogous to over-fitting in supervised learning.



(a) MDL Port-Graph (b) MDL IP-Graph

Figure 4.3: Shows the MDL for the DDCBM on the Port-Graph and the IP-Graph. The minimum value for the MDL is obtained for the model 54 groups in case of the Port-Graph and for 10 groups in case of the IP-Graph. The second smallest value of the MDL for the IP-Graph is obtained for 33 groups.

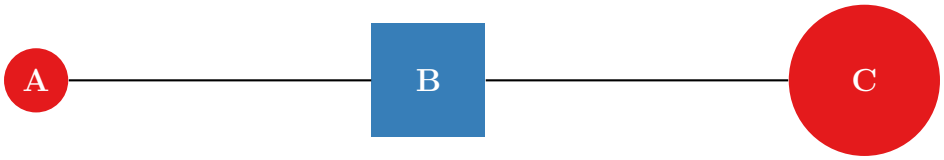


Figure 4.4: The vertex size indicates the degree and the color the membership assigned to nodes of this type.

In contrast, measures such as MDL strike a balance between model complexity and model quality, which is comparable to regularization.

4.2.2 Model Selection

Figure 4.3 shows the MDL for the DDCBM on the Port- and IP-Graph for values for $k = 2, \ldots, 100$. The minimum value of the MDL for the Port-Graph is obtained for $k = 54$ in Figure 4.3a and for $k = 10$ for the IP-Graph in Figure 4.3b. The second smallest value for the MDL on the IP-Graph is obtained for $k = 33$. For both graphs, the MDL shows roughly the same behavior. After initially high values, the MDL shows a decreasing trend until around $k = 50$ to $k = 60$ and then slowly starts to increase again. In case of the IP-Graph, the MDL has a sharp drop at $k = 10$ where the minimum value is attained. The Port-Graph also has a drop around this value, however not as strong. This drop indicates certain structural properties, which are captured for $k = 10$. The decreasing trend of the MDL for

small values of k and the increasing trend for larger values of k, can be interpreted similarly as the bias-variance trade-off in supervised learning. Simple models, that is small values for k, do not capture the patterns inherent in the data well enough, whereas larger values overfit. In case of the MDL, the minimum describes a good balance between model complexity and fit to the data.

With this in mind, the small value of $k = 10$ for the IP-Graph in Figure 4.3b indicates a far simpler structure than for the Port-Graph. A more thorough investigation of the split found for the IP-Graph reveals a specific structure that is identified by the DDCBM. Figure 4.4 illustrates this behavior. The figure shows three different vertex types. Type-A can be understood as periphery, vertices with very low degrees, which are connected to vertices of Type-B. Vertices of Type-B have total degrees of around 300 in the IP-Graph considered here, and in turn are mostly connected to vertices of Type-C, with huge total degrees of around 1 000. This structure can be mapped to a bipartite structure of two communities in which vertices of Type-A and Type-C are in one group and vertices of Type-B form the other group. This structure seems to describe the graph the best, but given the sharp increase and subsequent slow decrease of the MDL in Figure 4.3b this minimum might not yield a model that is complex enough to allow for meaningful results, i.e. the communities might be to coarse. Therefore, the local minimum at $k = 33$ might be more interesting.

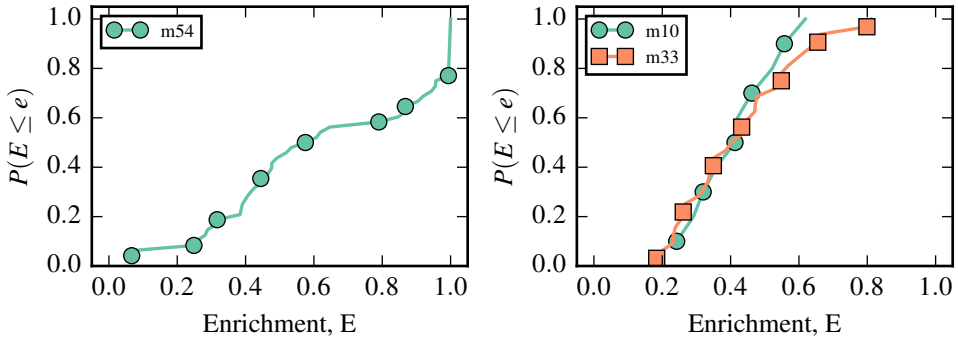
For the sake of completeness, the remainder of this chapter analyzes and compares the IP-Graph for the models with $k \in \{10, 33\}$ and for $k = 54$ in the case of the Port-Graph.

4.2.3 Enrichment

This section gives the enrichment for the groups found with the selected models, and combines them with the size of the inferred groups. The group size together with the enrichment values provide more detailed insight into whether the models split firstly, according to services and/or secondly, distinguish between client and server.

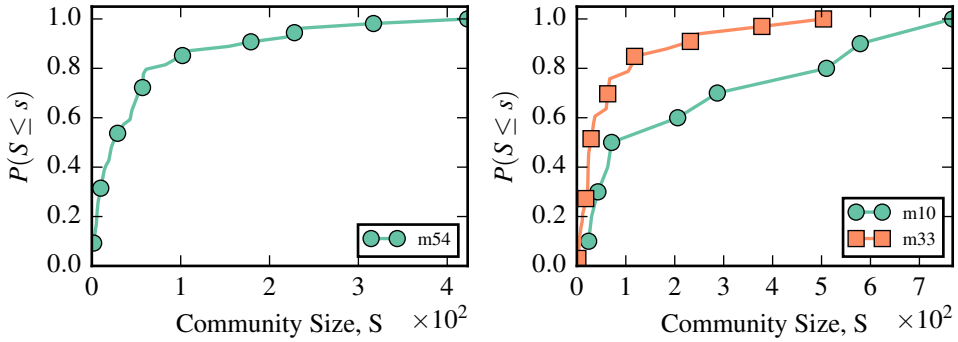
Figure 4.5 shows the cumulative distribution function (CDF)s for the models selected using MDL. The values are normalized to the interval $[0, 1]$ to make them comparable. For this, min-max-normalization is used with a minimum value of 0 and a maximum value of e_{bg}^{-1}, where e_{bg} is the similarity across all pairs of vertices. The choice of e_{bg}^{-1} can easily be seen in Equation (4.1). The nominator can maximally attain a value of one if all vertices have identical meta-data[2]. The interesting value of $e = 1$ is shifted for both graphs to 0.3. That is, the value 0.3

[2]This is not in general the case and only works because the used similarity measures have a maximum of one.



(a) Enrichment Port-Graph (b) Enrichment IP-Graph

Figure 4.5: Enrichment for Port- and IP-Graph obtained for the groups found by the optimal models described in the previous section. The maximum enrichment for the Port-Graph is 3.24 and for the IP-Graph 3.23.



(a) CDF of group sizes of the Port-Graph (b) CDF of group sizes of the IP-Graph

Figure 4.6: CDFs of group sizes for optimal models according to the MDL.

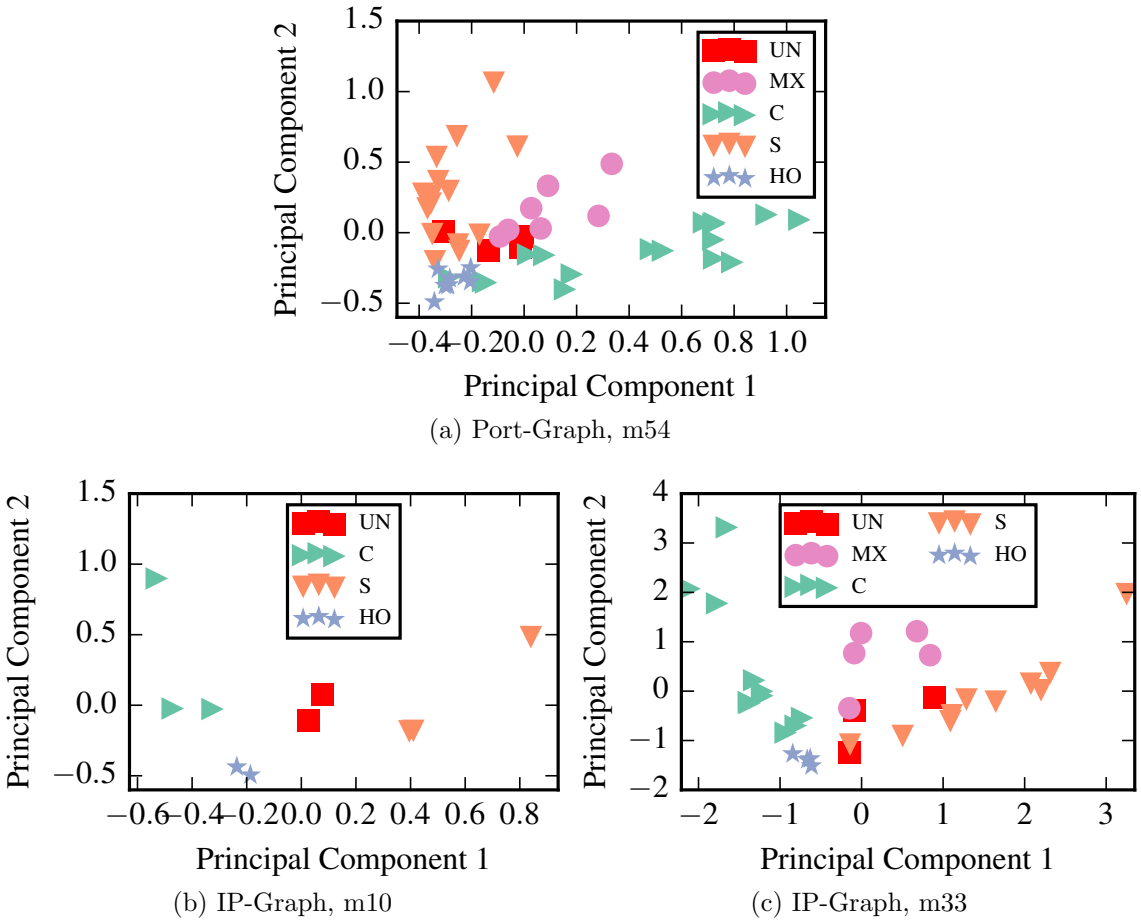
indicates in Figure 4.5 an equal distribution of services in a group compared to the distribution of services in the whole graph. Figure 4.6 shows CDFs of the group sizes of the selected models.

Port-Graph Figure 4.5a depicts the CDF of enrichment values of all groups for the selected model of the Port-Graph. As Figure 4.5a shows, around 20 % of the groups have normalized enrichment values of < 0.3 and contain thus vertices with different services. As discussed in Section 4.1.2, those groups might contain server. More than 20 % of the vertices have an enrichment value of 1 and contain thus vertices with the same service. Thus, groups for the Port-Graph in general do not correspond to single services.

But do enrichment values depend on the size of the community? If small groups mainly have high enrichment values and larger groups lower ones, this would indicate that the initial assumptions does not hold, i.e., hosts are not split into clients and servers, and high enrichment values are mainly due to sampling effects. If groups are split into clients and server, then larger groups, i.e., the clients are expected to be more homogeneous and thus have higher values for enrichment. Smaller groups i.e., servers are expected to be more diverse and thus have smaller values for enrichment. Figure 4.6a shows a wide range of group sizes from 1 to over 400, but most groups have sizes up to 100. The correlation coefficient between group size and enrichment is 0.22. This indicates that larger groups come with higher values for the enrichment. This also supports the initial assumption.

IP-Graph Figure 4.5b shows the CDFs for models with $k = 10$ and $k = 33$. For lower values of enrichment, both models show similar results. Over 40 % of the groups have near identical values. This indicates a similar composition of vertices in those groups for both models. The model with $k = 33$ achieves higher values of enrichment for the remaining groups. This is expected, as the model is more complex and thus is more likely to have higher scores as discussed in Section 4.2.1. Neither of the two models achieves a score of 1. Thus hosts in one group always use different services.

Around 70 % of the groups found by the model m33 are smaller than 100 vertices, as Figure 4.6b shows. In contrast, only 50 % of the groups found by model m10 are smaller than 100 vertices. This is only logical, since the model m10 has to accommodate the same number of vertices on fewer groups. Also, both models have a positive correlation coefficient between group size and enrichment. For model m10 the coefficient is 0.84 and for m33 0.41. Thus small communities tend to have smaller values for enrichment and larger groups larger values.



(a) Port-Graph, m54

(b) IP-Graph, m10

(c) IP-Graph, m33

Figure 4.7: Embedding of Groups into 2D-space by applying PCA to the features described in Table 4.3. *MX* refers to label *Mixed*, *C* refers to label *Client*, *S* refers to label *Server* and *UN* to *Unknown*.

Summary The general behavior of the DDCBM on the Port- and IP-Graph is similar. For both graphs exist groups with very heterogeneous and homogeneous vertices with respect to the services. Similarly, the group size correlates positively with the enrichment values for both graphs. Therefore the DDCBM primary identifies the roles, i.e. client and server, which might but need not correspond to single services. In general it is not possible to associate one specific group with one particular service.

4.2.4 Profiling Groups

The previous section gave indication into how groups relate to the roles of vertices (client vs. server) and the consumed services. This section further investigates the

separation into client groups and server groups. The goal is to clearly distinguish between client and server groups, and establish, whether servers offering the same or different services are grouped together. In order to discover this pattern, four features, i.e., the entropies of source and destination ports of incoming traffic (SRC-IN, DST-OUT) and outgoing traffic (DST-IN, DST-OUT), detailed in Table 4.3, are used. Figure 4.7 shows a two-dimensional embedding of the four features obtained for the groups of the models. The embedding is obtained using PCA. As Figure 4.7 shows, groups are embedded similarly for the Service- and the IP-Graph.

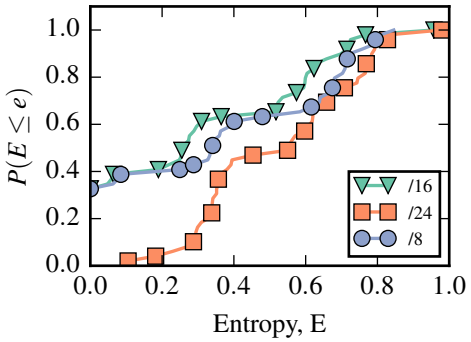
Client Groups are labeled as *C* and have high entropy values for SRC-OUT and DST-IN and low values for SRC-IN and DST-OUT.

Server Groups are labeled as *S* and have high entropy values for SRC-IN and DST-OUT and low values for SRC-OUT and DST-IN, i.e. just the opposite of the client groups. This is also reflected in Figure 4.7. Client and Server groups are farthest apart.

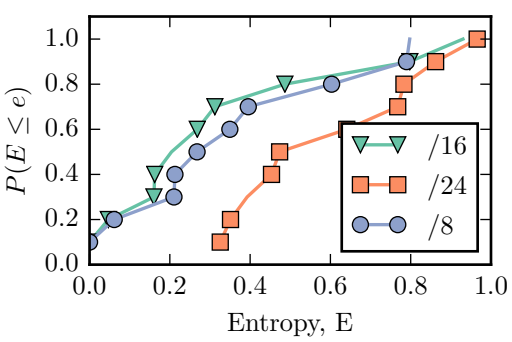
Mixed groups are labeled as *MX* and have high values for all four features. As Figure 4.7 shows, those groups occupy the space between the server and client groups. There are two primary reasons why server and client vertices are mixed together. Firstly, some server vertices look exactly like client vertices. For example, there are many HTTP server with only one client. That client has a rather large degree and also communicates with other client vertices with degree similar to the HTTP vertices. Thus the small degree clients and server are combined into one group, which is perfectly reasonable given only the graph.

Secondly, client and server vertices are grouped together that communicate mostly with a second group. That second group in turn contains the clients and server of the first group. Although reasonable in its own sense, it is not a desired behavior. A good split in this case would either be to separate the clients and the servers, or group the server together with their clients. The reason the DDCBM is not inferring one of the desired splits is rooted in the used heuristic algorithm to infer the communities. The algorithm gets stuck in a local optimum yielding that particular pattern.

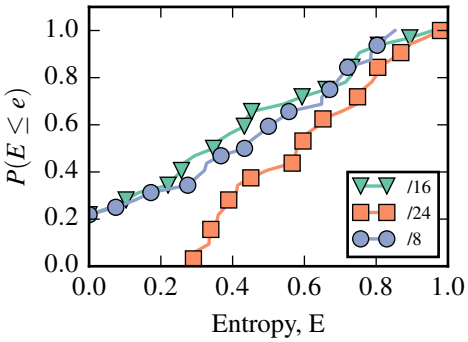
Homogeneous Groups are labeled as *HO*. The respective groups have low values for all four features. The groups contain server or clients and are associated with a service using a single source and destination port. Examples are NTP and NCP. A group labeled as *HO* can either be a server or client group.



(a) Port-Graph, model m54



(b) IP-Graph, model m10



(c) IP-Graph, model m33

Figure 4.8: CDFs of Entropy of IP-Subnets encountered in groups

Outliers are labeled as *UN* and refer to groups, that show neither of the above patterns. I.e., one feature has very high values while the others have low values or the other way round. Responsible for this behavior are services using unidirectional communication. All outliers can be traced to the Service Location Protocol (SLP), that uses User Datagram Protocol (UDP) multicast.

Summary For both, the IP-Graph and the Port-Graph the DDCBM yields similar results with respect to services and server-client roles. Most of the groups contain either server or clients. Some groups contain a mixture of server and clients of varying degree, that is some groups contain significantly more server vertices than clients and for some the ratio is about the same. Thus, hosts are separated firstly into server and clients vertices, and then according to who communicates with whom. This implicitly splits the hosts into groups with subsets of services, i.e., those services the respective hosts share.

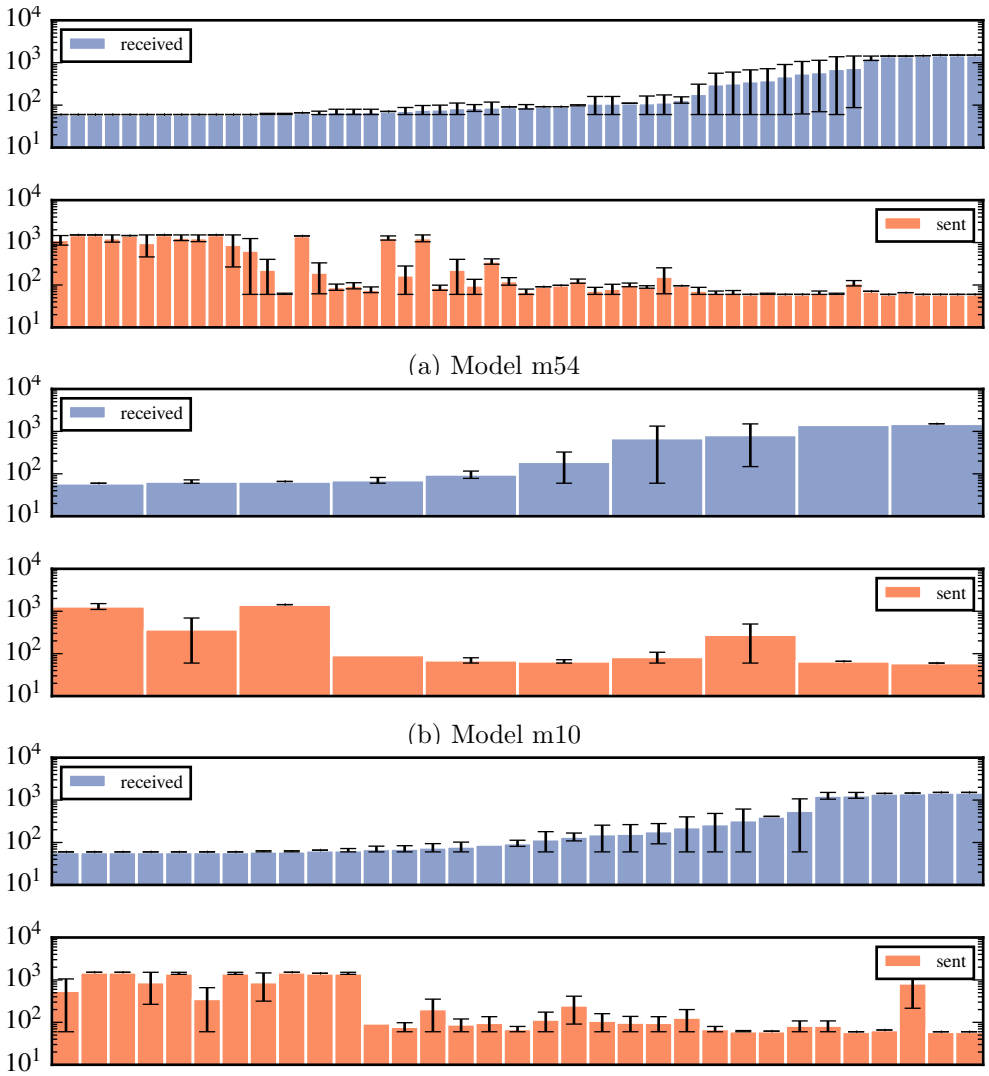
4.2.5 Correspondence of Groups to IP-Subnets

This section investigates, whether groups correspond to IP-Subnets. For this, Figure 4.8 shows the CDFs of m54, m33 and m10 for Class-A, Class-B and Class-C IP-Subnets. The entropy values are normalized to values in the interval $[0, 1]$ to make them comparable. Figure 4.8 shows that for all graphs and for all models groups do not correspond to single Class-C subnets. The number of groups that correspond to a single class-A or class-B subnet is the same for all considered models. For model m54 of the Port-Graph the number of pure groups is wit 38 % larger than for the models of the IP-Graph with 10 % and 20 % for m10 and m33 respectively. Model m10 has with 10 % the smallest number of pure groups. This is expected, as again only ten groups are available to group vertices into. Those results show that vertices are not grouped according to IP-Subnets in general. This makes intuitively sense given the earlier results. Groups containing hosts might be relative pure with respect to IP-Subnets whereas groups containing server are likely to contain many different subnets. For example there are groups that contain many HTTP-Server that are contacted from vertices inside the network. Those groups have a high range of different subnets. However, the results also show that most IP-Addresses fall into the same class-A subnets, since the curve of Class-A and Class-B subnets in all graphs are very similar.

4.2.6 Traffic running between Groups

This section seeks to answer whether the traffic, i.e. the package size for communication of different groups vary? Figure 4.9 shows the median values for the package sizes that are send and received from each group in log scale. Each bar corresponds to one group and the groups are sorted according to the median size of received packages for better illustration. The error bars represent the Median Absolute Deviation (MAD). The median and MAD are taken since the distribution of the package size is generally skewed, and thus mean and standard deviation not applicable [LLK+13]. For all models and graphs, Figure 4.9 shows varying values for the median package size. Some groups send and receive only small packages, while others send large- and receive small packages or vice verca. The only notable exception is the third last group in Figure 4.9a.

The large packages are caused by HTTP and SSH vertices. Interestingly, for the model m10 the package sizes already are quite distinguished across groups as Figure 4.9b shows. The groups from which large packages are mainly send contain server and are also contained in the groups labeled as server in Section 4.2.4. The groups receiving the large packages contain clients. Contrary to the server, the clients are not exclusively located in the group labeled as Clients in Section 4.2.4, but are also contained in all other groups.



(a) Model m54

(b) Model m10

(c) Model m33

Figure 4.9: Traffic running between communities. Plotted is the median. The lower bar represents the sent- and the upper bar the received median package size.

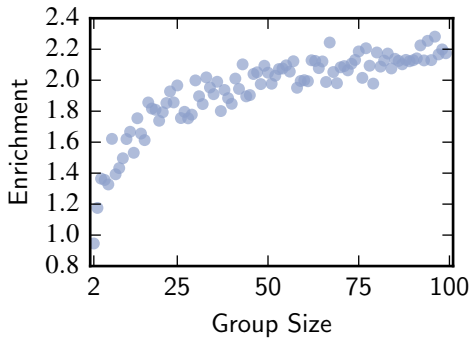


Figure 4.10: Average Enrichment of all DSBM models on the Port-Graph

The MAD indicates variability for some of the groups. But especially for m54 and m33, many groups express almost no variability in the size of send and received packages.

4.3 Evaluating Groups inferred with the DSBM

This section repeats the earlier evaluation for the Directed Stochastic Block Model (DSBM) to firstly investigate the capabilities of the DSBM for inferring meaningful groups, and secondly, see whether one of the two models is better suited to the task.

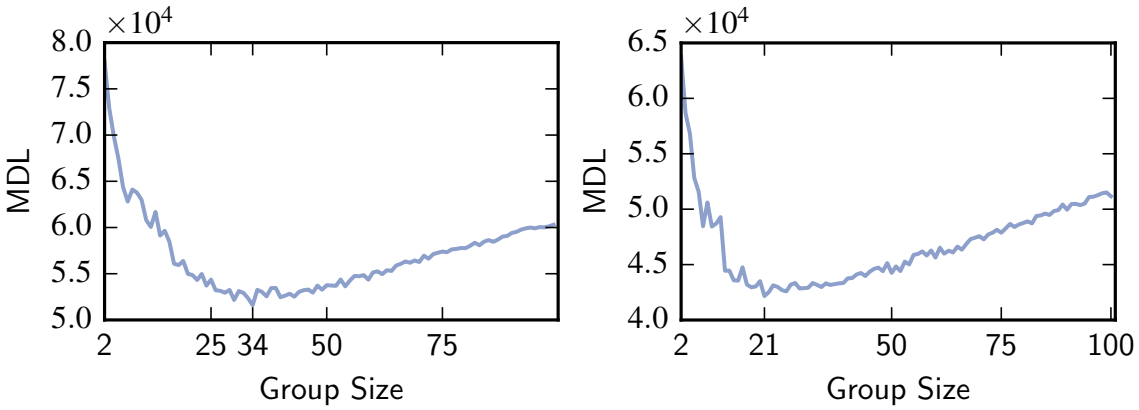
4.3.1 Influence of Parameter k

Figure 4.10 shows the CDF of the average enrichment for models with parameter $k = 2, \ldots, 100$. Similar to the DDCBM in Section 4.2.1 the average enrichment increases with the number of groups. Thus, a separate technique for model selection is needed for the DSBM as well.

4.3.2 Model Selection

Figure 4.11 shows the MDL for the DSBM on the y-Axis and the number of groups on the x-Axis. For the Port-Graph the minimum MDL is attained for $k = 34$ groups as Figure 4.11a shows. For the IP-Graph the minimum MDL is attained for $k = 21$ groups visible in Figure 4.11b. This indicates a more simple structure which is reasonable given that the vertices of the Port-Graph are collapsed into as single vertex in the IP-Graph.

The group sizes for the DSBM on the Port-Graph is far smaller than for the DDCBM with $k = 34$ compared to $k = 54$. Thus, the DSBM compresses the



(a) MDL Port-Graph (b) MDL IP-Graph

Figure 4.11: Shows the MDL for the DSBM on the Port-Graph and the IP-Graph. The minimum value for the MDL is obtained for the 34 groups in case of the Port-Graph and for 21 groups in case of the IP-Graph.

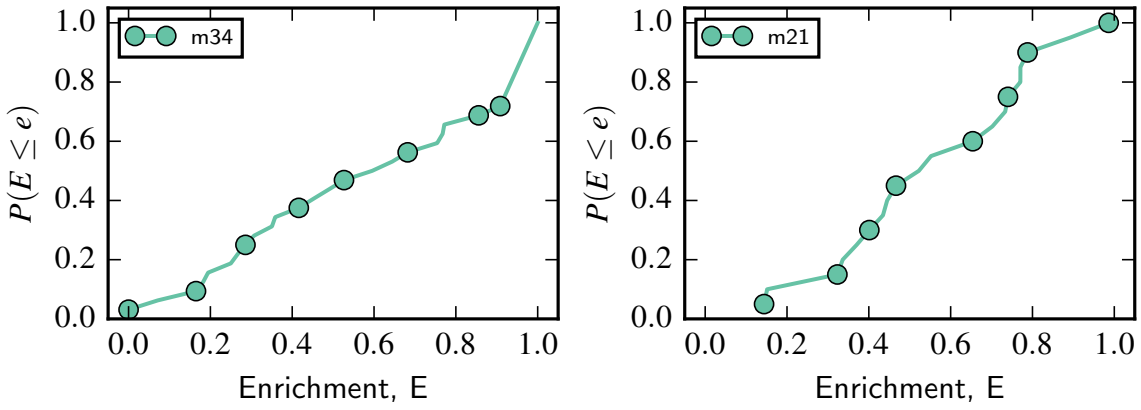
graph more than the DDCBM. For the IP-Graph it is vice verca, here the DDCBM needs only $k = 10$ groups compared to $k = 21$ for the DSBM. The previous section showed that the respective groups are rather coarse and mix server with client vertices. The next smallest MDL value for the DDCBM is then $k = 33$, which again is larger than the $k = 21$ groups for the DSBM. Leaving the $k = 10$ for the DDCBM aside, the DSBM needs less groups than the DDCBM to describe the graph.

4.3.3 Enrichment

Figure 4.12 shows the CDFs for the enrichment of the model m34 for the Port-Graph and m21 for the IP-Graph. Figure 4.13 shows the respective group sizes.

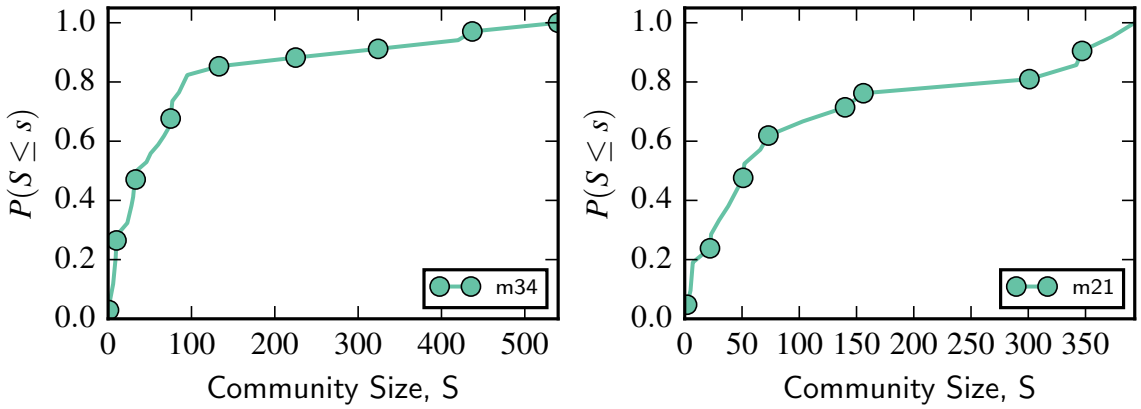
Port-Graph Figure 4.12a shows the existence of a small number of groups with enrichment $E = 0$, i.e., groups in which no vertices have the same service. About 20 % of the groups have an normalized enrichment value smaller than 0.3 (the value 0.3 of the normalized enrichment corresponds to $E = 1$ in the un-normalized case, i.e., groups with a value of 0.3 are equally diverse as the graph itself) and are thus more heterogeneous than the baseline. The remaining groups contain vertices with meta-data that is more similar than the baseline. Around 20 % of those attain the maximum value of one.

Figure 4.13a shows the CDF of group sizes for m34. The Figure shows that over 80 % of the groups has less than 100 vertices, and about 20 % are very small



(a) Enrichment Port-Graph (b) Enrichment IP-Graph

Figure 4.12: Enrichment for Port- and IP-Graph obtained for the groups found by the optimal models described in the previous section. The maximum enrichment for the Port Graph is 3.24 and for the IP-Graph 3.22



(a) CDF of group sizes of Port-Graph (b) CDF of group sizes of IP-Graph

Figure 4.13: CDFs of group sizes for optimal models according to the MDL

with less than 20 vertices. The check whether small group sizes come with large values for enrichment, the correlation coefficient between group size and enrichment value is calculated. Similar to Section 4.1.2, small groups are expected to contain server and should thus have small values of enrichment. The correlation coefficient between group size and enrichment is 0.27. This indicates a tendency of larger groups to also have larger values for enrichment, which is what is expected.

IP-Graph For the IP-Graph less than 20 % of the groups are more heterogeneous than the baseline. Figure 4.12b illustrates this, and depicts the CDF of enrichment values for the groups of m21. Contrary to the Port-Graph, no group has a value of zero, i.e., in all groups some vertices share services. The same is true for the maximum enrichment. No group exists in which all vertices are associated with the same set of services.

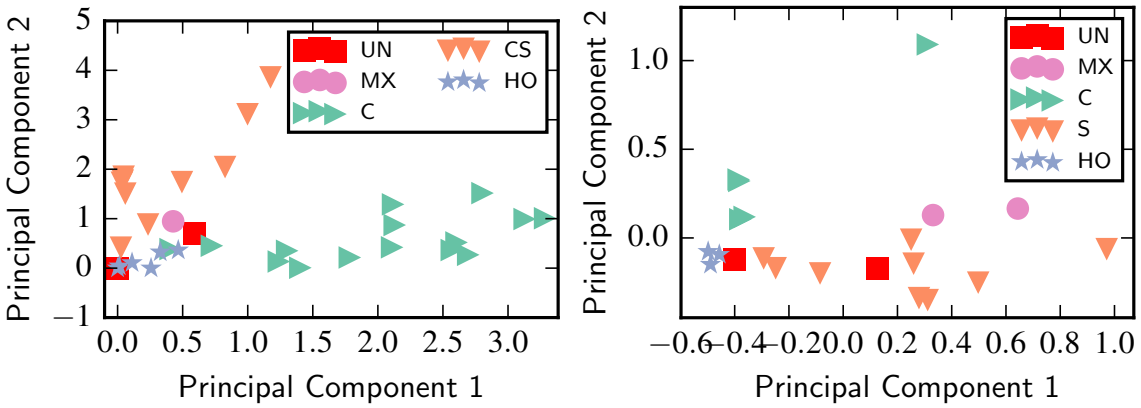
Regarding the group sizes, the CDF of group sizes in Figure 4.13b shows that about 20 % of the groups are very small, i.e., less than 10 vertices. Around 60 % of the groups contain less than 100 vertices. The largest group contains 350 vertices, which is 150 vertices less than the largest group of m34. Group size and enrichment also correlate positively with 0.13, which is less than for the Port-Graph, but still indicates heterogeneous, smaller groups.

Summary The results are similar to those of the DDCBM. Vertices are primary split according to their role in the graph. As the enrichment values show, the result is an implicit split according to services. But just as with the DDCBM do single groups inferred with the DSBM not correspond to one single service, but rather to a mix of services for which associated vertices behave in the same fashion. The DSBM is more rigorous in this respect, as there exist groups with completely dis-similar vertices for the Port-Graph, contrary to the groups inferred with the DDCBM.

4.3.4 Profiling Groups

Figure 4.14 shows the embedding of the port characteristics detailed in Table 4.3 into two dimensional space. Figure 4.14a shows the embedding for m34 and Figure 4.14b for m21. In both cases, groups containing server and groups containing clients can be distinguished. For both graphs groups exist that have very low values for all four features. Those are labeled as *HO*. They again correspond to groups containing e.g. NTP hosts and servers.

Mixed groups are inferred by both models and as in Section 4.2.4 indicate a mix of server and clients. In case of the Port-Graph, this happens because some



(a) Embedding of Groups for Port-Graph (b) Embedding of Groups for IP-Graph

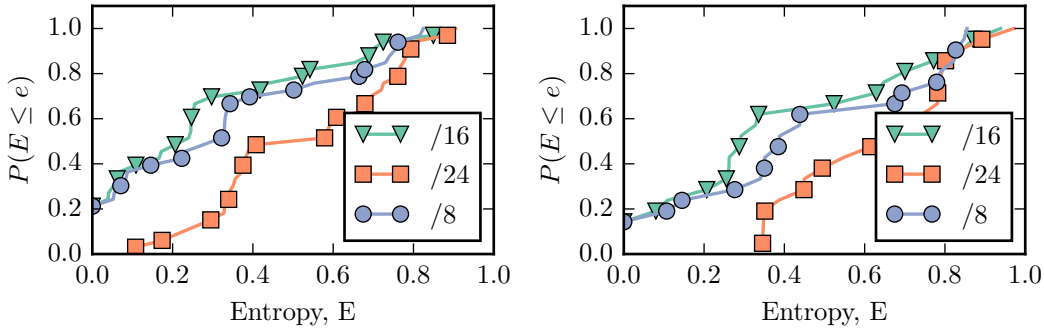
Figure 4.14: Embedding of Groups into 2D-space by applying PCA to the features described in Table 4.3. *MX* refers to label *Mixed*, *C* refers to label *Client*, *S* refers to label *Server* and *UN* to *Unknown*.

server vertices share the same behavior as client vertices. That is, there are some vertices with larger degree in the graph that are labeled as clients (i.e. have a port larger 1024). They communicate on the one had with other client vertices and on the other with server vertices that both have a smaller degree. Thus the larger client takes the role of a *server* for the smaller clients and server, which are both identified as *clients* of the *server*.

Another pattern yielding high values in all four features is caused by combining server with their clients in the same group. The server connects only to clients of that group, but the clients also have connections to vertices of other groups. Both, the client and the server have similar degree due to the clients additional connections. Thus this split is reasonable given only the graph without additional information.

Two such groups are identified for the IP-Graph and one group for the Port-Graph. This is far less compared to Figure 4.7. The DSBM thus better distinguishes between servers and clients.

Outlier The groups of the Port-Graph that have un-symmetric feature pattern contain vertices using SLP. The protocol includes UDP multicast which causes the observed imbalance in the feature. The observed outliers for Port- and IP-Graph are no all related to unidirectional protocols. The DSBM split the vertices using SLP differently compared to the DDCBM. This results in groups with well behaved values for the features.



(a) Port-Graph, model m34

(b) IP-Graph, model m21

Figure 4.15: CDFs of Entropy of IP-Subnets encountered in groups

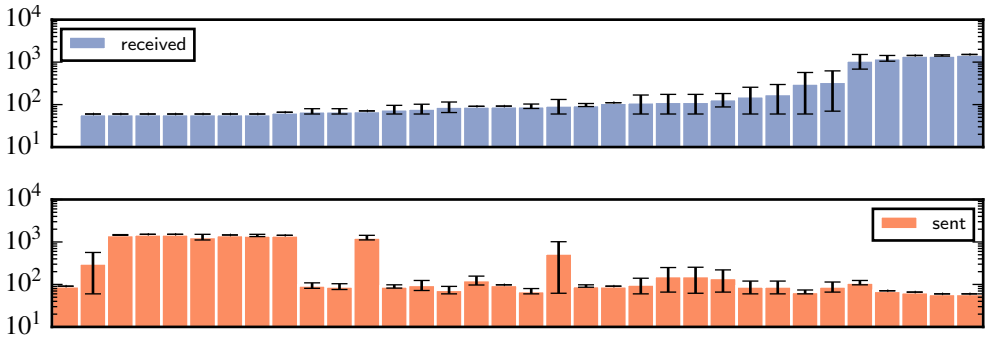
Summary As before the results on the Service- and IP-Graph are similar. In both cases vertices are split with respect to their role. Some groups can be separated into those containing clients, server or a mix between them. Contrary to mixed groups inferred with the DDCBM, mixed groups inferred with the DSBM on the Port-Graph arise solely due to the fact, that some server look like clients given the graph alone. Thus, the groups inferred with the DSBM capture the different roles of the vertices better.

4.3.5 Correspondence of Groups to IP-Subnets

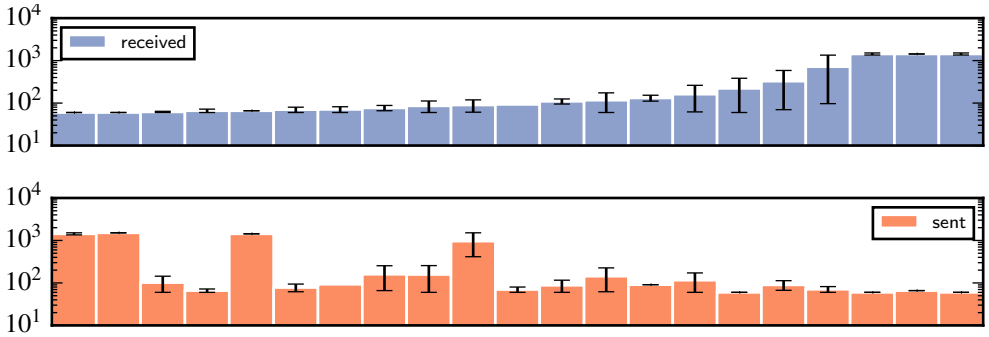
Figure 4.15a shows the CDF of the entropies of Class-A, Class-B and Class-C subnets for the groups inferred with m34 on the Port-Graph, and Figure 4.15b respectively for the IP-Graph and model m21. The entropy values for Class-A and Class-B subnets behave similarly for both graphs, i.e. around 20 % of the groups are pure. For Class-C nets are groups inferred on the Port-Graph purer than on the IP-Graph. For the Port-Graph the smallest entropy value is 0.1 and for the IP-Graph 0.36 for the Class-C nets. In general, the same results as for the DDCBM is obtained, i.e. there is no general correspondence between IP-Subnet and group.

4.3.6 Traffic running between Groups

Figure 4.16a shows the median size of send and received packages of vertices in each group for m34 on the Port-Graph. Error-bars depict the MAD. Figure 4.16b respectively for m21 on the IP-Graph. The figures highlight the separation into client and server groups, i.e. traffic mostly runs *between* groups and not *within* groups. This ins indicated by similar bar heights for received and sent packages



(a) Model 34



(b) Model m21

Figure 4.16: Traffic running between communities. Plotted is the median. The lower bar represents the sent- and the upper bar the received median package size.

sizes, i.e., one group sends, the other one receives. The results are similar to those obtained with the DDCBM, some groups send jumbo packages, while other send only small packages.

4.4 DDCBM vs. DSBM

After investigating DDCBM and DSBM separately on the Port- and IP-Graph, compares this section DDCBM and DSBM directly with each other. Section 4.4.1 considers performance on the Port-Graph, and Section 4.4.2 on the IP-Graph.

4.4.1 Port-Graph

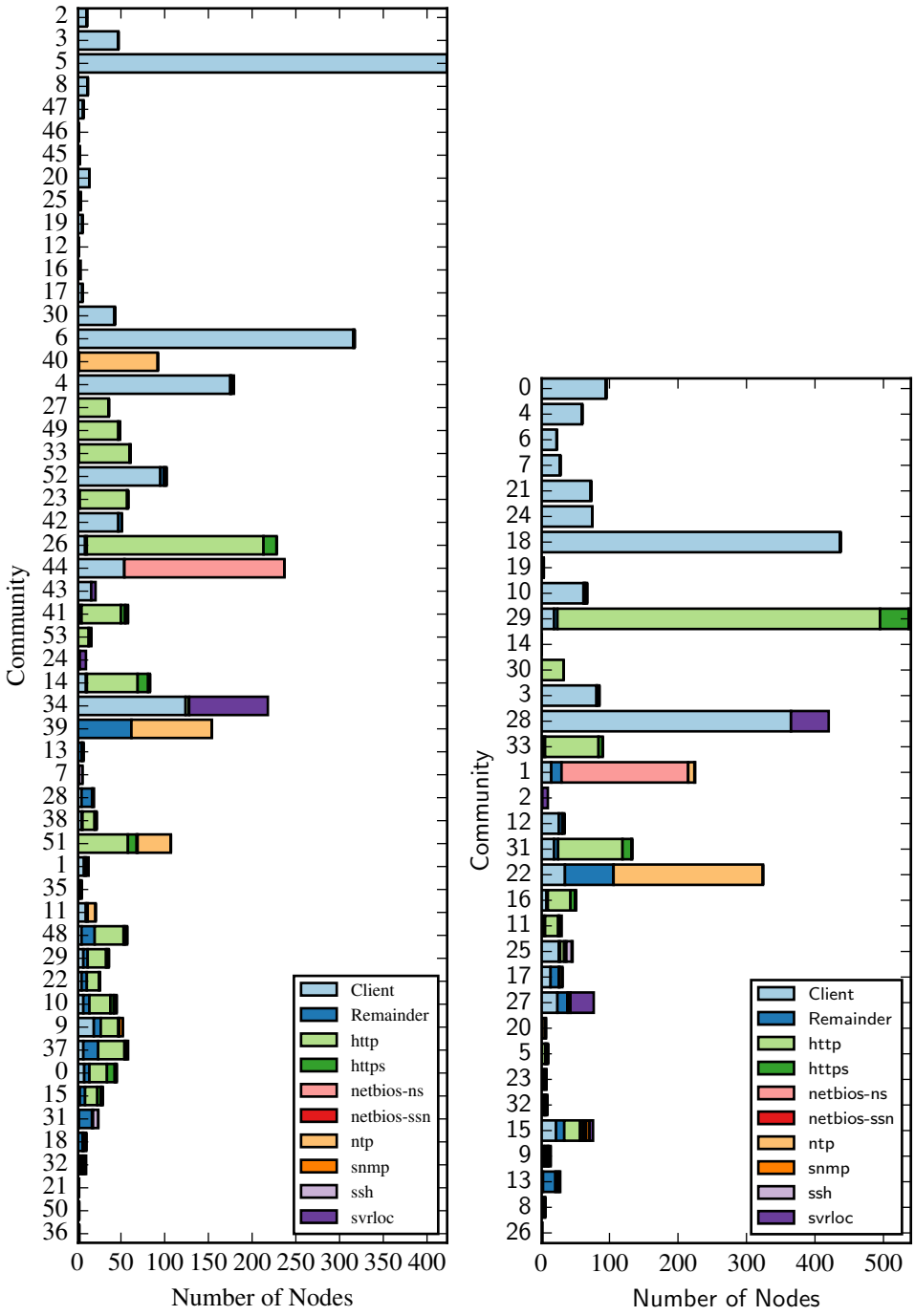
Figure 4.17 shows a stacked bar chart illustrating how the top nine most important services are distributed in the inferred groups for models m54 and m34 inferred with the DDCBM and DSBM respectively. The x-Axis depicts the number of vertices in each group. The y-Axis corresponds to single groups. Groups are sorted descending according to their enrichment score, that is, top groups have highest enrichment, and bottom groups lowest. The color of each bar indicates the service. Figure 4.17 shows that both models separated client vertices from server vertices. Also, both models grouped hosts providing different services together.

The split between servers and clients seems to be better for the DDCBM. Groups inferred with the DSBM contain more frequently a mix of hosts and servers. This can often be explained by servers looking like clients given only the graph. For example vertices in group 31 in Figure 4.17b are all low-degree vertices and connect to the same host. Given only the graph and no additional information they cannot be differentiated.

The more homogeneous groups of the DDCBM are offset by the high number of groups that have only one or two vertices. Especially, since those groups account for many of the high enrichment groups. The separation of the DSBM also contains small groups, however, they have mostly low enrichment scores, i.e., they contain servers.

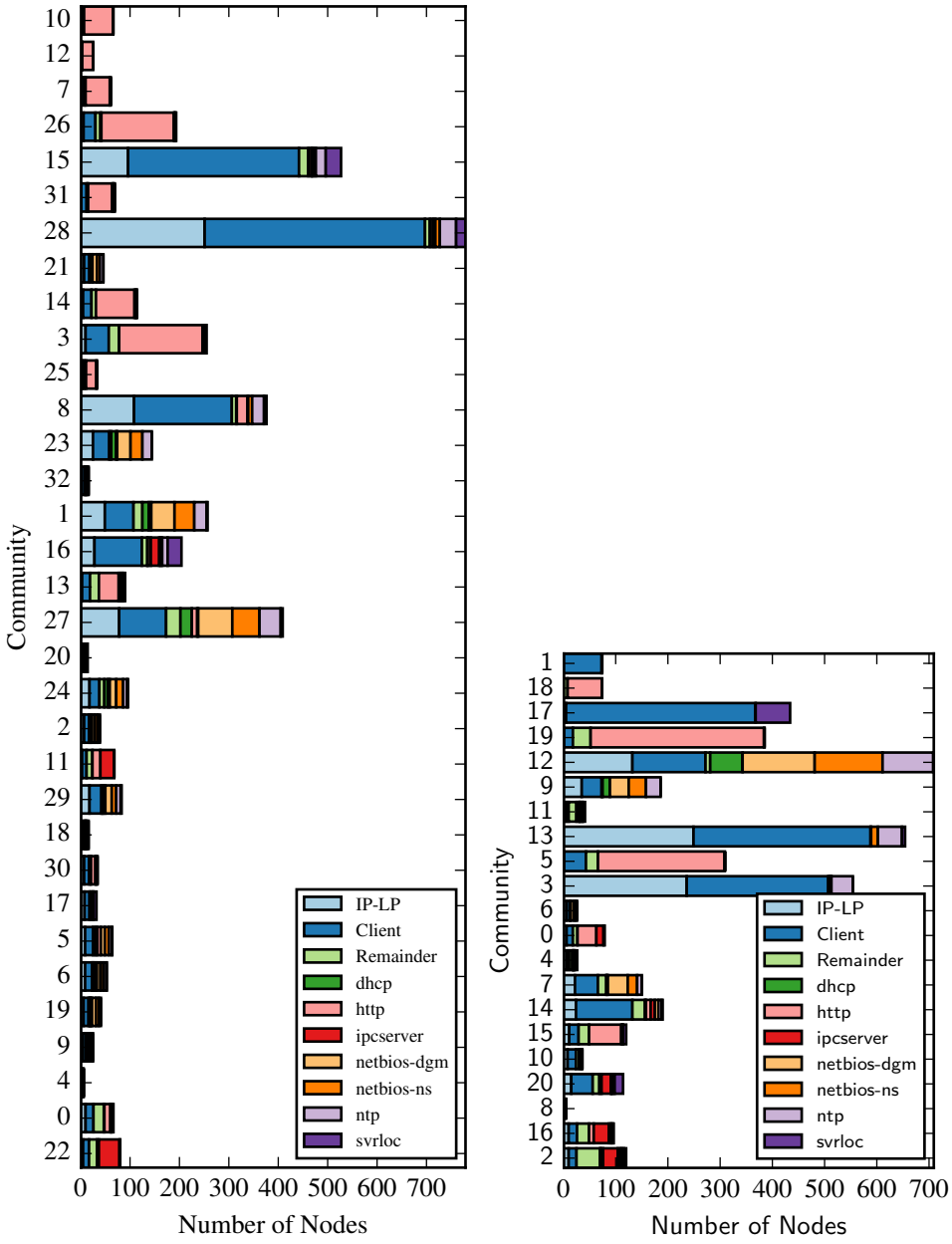
4.4.2 IP-Graph

For the IP-Graph, only models m33 and m21 for DDCBM and DSBM respectively are considered. Based on the insights of previous sections, m10 of the DDCBM is deemed not suited to represent meaningful groups. Figure 4.17 shows a stacked bar plot of the most frequent services in the IP-Graph. The color of the bars gives the services associated with respective clients. The composition of services in one group gives co-occurring services for vertices, i.e., each vertex is associated



(a) Port-Graph, DDCBM (b) Port-Graph, DSBM

Figure 4.17: Most important services per group for Port-Graph. Each figure shows the top nine most frequent services in the Port- Graph for groups inferred with the DDCBM and DSBM respectively. The groups are sorted according to enrichment descending, that is, the top group has the highest enrichment and the bottom group lowest enrichment. The y-Axis shows the size of groups. The bars with label *Remainder* contain the sum of the remaining vertices with services that are not among the top nine.



(a) IP-Graph, DDCBM (b) IP-Graph, DSBM

Figure 4.18: Most important services per group for the IP-Graph. Each figure shows the top nine most frequent services in the IP-Graph for groups inferred with the DDCBM and DSBM respectively. The groups are sorted according to enrichment descending, that is, the top group has the highest enrichment and the bottom group lowest enrichment. The y-Axis shows the size of groups. The bars with label *Remainder* contain the sum of the remaining vertices with services that are not among the top nine.

with multiple services. The width of each bar does therefore not directly translate to the number of vertices in a group, since vertices might be counted multiple times. The y-Axis represents groups, groups are sorted descending according to their enrichment score.

Small groups have low enrichment scores for both models. Thus, both models group different servers together. Also, both models separate clients into few groups. Larger groups inferred with the DSBM are more homogeneous compared to those of the DDCBM. For example HTTP is mostly concentrated into groups 18, 19 and 5 for the DSBM. For the DDCBM, most of the vertices associated with HTTP are spread over seven groups.

4.4.3 Summary

Both models are able to separate clients and servers, and implicitly group vertices according to associated services. That is, hosts consuming the same services from the same servers and the respective servers form one group each. In summary, the DSBM is superior the the DCBM. The DSBM better separates between servers and clients, and yields similar or better results regarding enrichment, i.e., purity of the groups, as the DDCBM.

4.5 Summary

This chapter illustrates how the groups inferred with the DDCBM and DSBM on the Port- and IP-Graph relate to services associated with the vertices. Both models split the vertices of both graphs primary according to their role. That is, groups contain either servers or clients. Groups containing vertices of both roles arise in the context of the DDCBM on the Port-Graph due to servers that behave like hosts, but also due to the objective function of the model and the used heuristic algorithm. For the DSBM mixed groups arise solely because of servers acting like hosts. In the IP-Graph, some groups exist that contain servers, which themselves consume other services. This then also results in groups with both, client and server characteristics.

Single groups do in general not correspond to single services, but rather to multiple services for which the associated vertices show the same behavior with respect to their connection pattern. Thus, an implicit separation takes place.

For all graphs and models, the inferred groups do not in general correspond to IP-Subnets of any class.

The median size of packages send between groups varies greatly. Some groups send many jumbo packages, while others send only small packages. This revelation might be leveraged for dynamic workload creation or routing of flows.

Finally, this chapter shows that the DSBM is better suited for the grouping of vertices for this type of graph. The DDCBM tends to mix server and clients that form a bipartite structure. While this nicely explains the structure of the graph, it is not suited to extract groups that are meaningful in the sense of services and host/server roles.

Chapter 5

Topology Generation

This chapter investigates the capabilities of the presented generative models to reproduce graphs, and is organized as follows, Section 5.1 discusses the difficulties in generating- and desired properties of synthetic graphs. Section 5.1 presents the empirical graphs, which should be reproduced, as well as the graph properties, being used to assess the quality of the generated graphs. Section 5.1 also describes how the probabilistic models are used to generate synthetic graphs, and introduces a reference generator. Section 5.3 then analyzes the generated graphs, and compares them to those obtained with the reference generator. Section 5.4 closes the chapter by summarizing the results and giving an outlook.

5.1 Problem Statement

Synthetic graphs are used to design, implement and validate algorithms for different networking applications [RFT13]. Synthetic graphs should closely resemble properties of real world networks, since they indicate certain topological properties, which heavily influence the performance and behavior of algorithms and applications [RFT13, MKF+06]. The topology of graphs arising in distinctive areas is different. The graph formed by the IP-to-IP communication between hosts is different than the underlying physical router graph. Many of the existing topology generators are specifically tailored to the graph of the Internet, i.e., the Autonomous System (AS) or router-level topology. The presented probabilistic models are more generic in that they make no assumptions about the underlying structure. This chapter uses the probabilistic models to reproduce the an router-level graph, and the IP-to-IP communication graph introduced in the previous chapter.

The problem tackled in this chapter is thus the generation of synthetic graphs with a similar topology as the respective empirical graphs. The graphs are also

required to express some variability, i.e., it is *not* the goal to exactly reproduce the empirical graphs. Optimally, the parameters of the probabilistic models should allow control over properties of the resulting synthetic graphs. A challenge with this respect is the choice of the free parameter k. As in the previous chapter, MDL is used to indicate possible good values. The final choice is made based on visual inspection and domain knowledge, though.

Similarity and variability of synthetic graphs are assessed by calculating different graph properties for each synthetic graph, and compare them to the respective original values. This approach is frequently used [RFT13][AGL15, MKFV06, MKF+06, MMB00].

5.2 Material and Methods

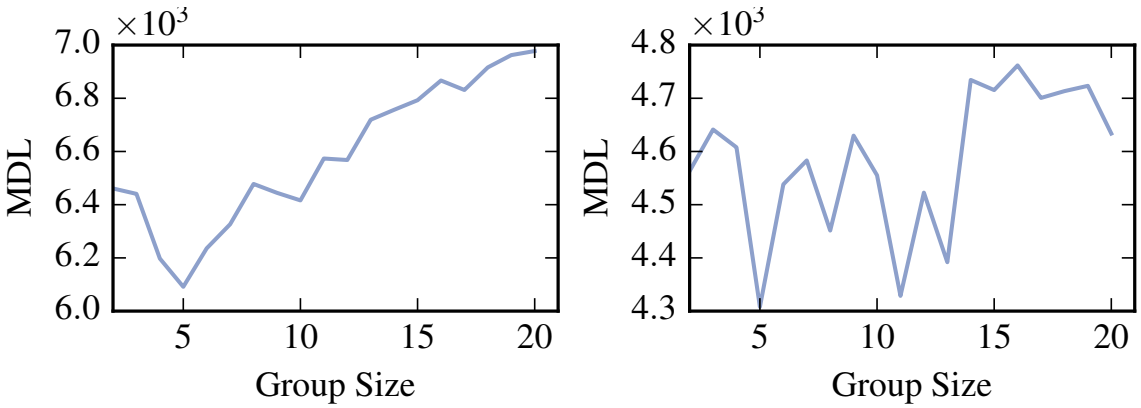
The used graph are presented in Section 5.2.1 and key differences discussed. Section 5.2.2 outlines how different capabilities of the models can be combined to compensate for deficiencies, and Section 5.2.3 how free parameters for the models are chosen. Section 5.2.4 then describes how synthetic graphs are generated with each models, and Section 5.2.5 then introduces the reference generator Orbis.

5.2.1 Used Graphs

The router level graph used in this chapter is the HOT graph obtained from the webpage of the authors of Orbis [Mah]. The HOT graph is a simple *undirected* graph, therefore, SBM and DCBM are used to generate synthetic graphs. The IP-to-IP communication graph is the LBNL graph from the previous chapter. Due to being directed, the DSBM and DDCBM are used to generate synthetic graphs. Table 5.2 and Table 5.3 give values for different graph attributes. The chosen graph attributes are frequently used to assess topology generators, and encode topological structure with impact on networking algorithms and applications [MKFV06, RFT13, AGL15, MMB00]. The remainder of this section outlines the differences between the graphs based on the clustering coefficient, shortest path and degree assortativity coefficient. The selected properties indicate topological structure with a large impact on network applications [MKF+06].

The clustering coefficient is 0 for the HOT, and 0.1 for the LBNL graph. Taking the $G(n, p)$ model as null model, the expected clustering coefficient for the HOT graph would be $1.85e^{-3}$ and $2.24e^{-3}$ for the LBNL graph. The LBNL has more triangles compared to the HOT or a random graph. The formation of triangles (i.e. cliques) is known to affect routing algorithms [MKF+06].

Both graphs are similar in having a negative degree assortativity coefficient. However, the coefficient of the HOT graph is with -0.22 larger than the coefficient



(a) MDL for SBM (b) MDL for DCBM

Figure 5.1: MDL for SBM and DCBM on HOT graph.

of the LBNL graph with -0.31. This indicates that in both graphs low degree vertices connect to high degree vertices. For the LBNL graph this behavior is more pronounced as indicated by the smaller value of -0.31. Both graphs are thus vulnerable to targeted attacks or random vertex failures. On the other hand, prevention of Distributed Denial of Service (DoS) attacks or traffic monitoring is more efficient, since the negative degree assortativity coefficient indicates that few vertices are incident to many others [MKF+06].

The average path length of the HOT graph is with 6.81 significantly larger than for the LBNL graph with 2.87, although the LBNL graph has 2.5 times more vertices. The respective values based on the $G(n, p)$ model would be 2.97 for the HOT and 3.41 for the LBNL graph. It is easy to see how this property would impact routing algorithms,

In summary, the LBNL graph is stronger connected, has stronger community structure and has considerably shorter paths than the HOT graph, and should be reflected in the synthetic graphs as well.

5.2.2 SBM+DCBM

SBM+DCBM uses the SBM or DSBM for inference and the DCBM or DDCBM for generation. This model infers the same split as the SBM and is able to reproduce heterogeneous degrees within groups, which aids in reproducing the heavy tail observed in the degree distributions.

5.2.3 Model Selection

As before, the number of groups k must be chosen for the HOT and LBNL graph respectively. MDL values are calculated for models with different values for k. For each k, parameters z, M, θ and $\theta^{in}, \theta^{out}$ are fitted ten times and the parameters yielding the largest likelihood are taken. The remainder of this section discusses how the value k is then chosen for each model.

LBNL graph According to the previous chapter, the minimum MDL is obtained for the DDCBM on the LBNL graph for $k = 10$ and for the DSBM for $k = 21$ groups in Figure 4.3b and Figure 4.11b respectively. In the subsequent analysis $k = 10$ is used for the DDCBM and DSBM. The use of different values of k in Chapter 4 is justified, since Chapter 4 is concerned with the quality of the inferred groups. In contrast, this chapter looks into the capabilities of DDCBM and DSBM to generate realistic synthetic graphs. Thus, only the capability to capture the *structure* of the graph is of interest, i.e., the value of k depends on the application. A smaller number of groups imposes less restrictions on the resulting graphs, as a larger number of groups. As a consequence, more distinctive graphs can be drawn from the model. Put differently, the probability of creating a graph with the same topology twice is reduced. To assess whether the DDCBM or DSBM better reproduces the LBNL graph using the same number of groups, graphs generated from the DDCBM and DSBM with $k = 10$ are evaluated.

HOT graph The minimum value for the MDL is attained for $k = 5$ for the DSBM as well as the DDCBM. Figure 5.1 shows the MDL values for both models for $k = 1, \dots 20$ respectively. Nevertheless, a value of $k = 4$ is chosen for the subsequent analysis, after visually inspecting the respective result.

5.2.4 Synthetic Graph Generation

This section shows how synthetic graphs can be efficiently generated from SBM, DSBM, DCBM and DDCBM. The so generated graphs are not guaranteed to be connected. Therefore, a post-processing routine is proposed that, under certain conditions, transfers the unconnected graph to a connected graph. This section is structured as follows, firstly, efficient graph synthesis is explained. Secondly, conditions for generating a connected graph are given. Thirdly, the post-processing routine transforming an unconnected graph, which fulfills the previously given constraints, into an connected graph is explained. Finally, the section closes by explaining how simple directed or undirected graphs are obtained. A summary over symbols used in this section is given in Table 5.1.

Algorithm 2 Algorithm merging unconnected components of a graph generated with a SBM or DCBM. The outer while loop is active as long the graph is either not connected or an merge occurred during one iteration. If no merge occurred and the graphs is still unconnected, it is impossible to obtain a connected graph and a new one must be generated with different seed.

while (not connected) and (merge occured) **do**
  for each component $C = (\mathcal{V}^c, \mathcal{E}^c)$ **do**
    $\mathcal{V}^{c\prime} := \mathcal{V}^c$
    while $\mid \mathcal{V}^{c\prime} \mid > 0$ **do**
      $v :=$ choice$(\mathcal{V}^c)$
      $\mathcal{Q} := \{w \in \mathcal{V}/\mathcal{V}^c \mid z_w = z_v \wedge d(w) > 1\}$
      if $\mid \mathcal{Q} \mid > 0$ **then**
        break
      else
        $\mathcal{V}^{c\prime} := \mathcal{V}^{c\prime}/\{w \in \mathcal{V}^{c\prime} \mid z_w = z_v\}$
      end if
    end while
    if $\mid \mathcal{Q} \mid > 0$ **then**
      while true **do**
        q := choice$(\mathcal{Q})$
        $\mathcal{N}'_q := \{m \in \mathcal{N}_q \mid d(m) > 1\}$
        if $\mid \mathcal{N}'_q \mid > 0$ **then**
          n = choice$(\mathcal{N}'_q)$
          $\mathcal{E} := \mathcal{E}/\{(q, n)\}$
          $\mathcal{E} := \mathcal{E} \cup \{(v, n)\}$
        else
          $\mathcal{Q} := \mathcal{Q}/\{q\}$
          if $\mid \mathcal{Q} \mid = 0$ **then**
            break
          end if
        end if
      end while
    end if
  end for
end while

Symbol	Meaning
$\mathcal{V}_r$	Vertices belonging to group r.
$C = (\mathcal{V}^c, \mathcal{E}^c)$	One component of an unconnected graph.
$\mathcal{E}_r s$	All edges running between groups r and s.
$\mathcal{E}_r$	All edges incident to vertices in group r.
$\mathcal{Q}$	Set of vertices induced by a vertex v of in an unconnected component $\mathcal{V}^c$.
λ_{e_s}	Mean value of $\mid \mathcal{E}_s \mid$.
$\mathbb{I}_{r \neq s}$	Indicator function, 1 if $r \neq s$ and 0 else.

Table 5.1: Overview of symbols used throughout this section.

Efficient Graph Synthesis Graphs arising in the context of communication networks are usually sparse. For instance, LBNL and HOT have a density of only $1.85e^{-3}$ and $2.24e^{-3}$ respectively. This property can be used to speed up graph synthesis, i.e., to avoid a runtime quadratic in the number of vertices. Let $X_1 \sim \text{Poi}(\lambda_1), \ldots, X_n \sim \text{Poi}(\lambda_n)$ be independently and identically distribute (iid) RVs, then their sum $Z = \sum_{i=1}^{n} X_i$ is distributed as $Z \sim \text{Poi}(\sum_{i=1}^{n} \lambda_i)$. As stated in Section 2.4, edges $\mathcal{E}_{rs}$ between two groups r and s are iid Poisson RVs. The number of edges $\mid \mathcal{E}_{rs} \mid$ is thus also a Poisson RV. Using this fact, graphs are generated in three stages. Firstly the number of edges $\mid \mathcal{E}_{rs} \mid$ is generated as:

$$\mathcal{E}_{rs} \sim \begin{cases} \text{Poi}(\frac{1}{2} n_r \mid \mathcal{V}_r \mid\mid \mathcal{V}_s \mid M_{rs}) & \text{if } r = s \text{ and graph undirected} \\ \text{Poi}(\mid \mathcal{V}_r \mid\mid \mathcal{V}_s \mid M_{rs}) & \text{else.} \end{cases} \tag{5.1}$$

Secondly, for each edge, a head and tail is drawn from the vertices in groups r and s respectively. For the SBM, vertices in one group are stochastically equivalent. Every vertex is thus equally likely chosen as head or tail. Let $v \in \mathcal{V}_r$ and $w \in \mathcal{V}_s$, the probability of v to be chosen as tail is then $\frac{1}{|\mathcal{V}_r|}$, and the probability of w to be chose as head $\frac{1}{|\mathcal{V}_s|}$. For the DCBM, respective probabilities are proportional to θ_v, θ_w, i.e. $\theta_v^{out}, \theta_w^{in}$ for the DDCBM. The resulting graph might be unconnected. Therefore, the third step rewires edges to obtain a connected graph.

Assumptions underlying the merging Algorithm In order to generate a connected graph $G = (\mathcal{V}, \mathcal{E})$, the number of edges incident to a group s given by:

$$\mathcal{E}_s = \begin{cases} \bigcup_{r=1}^{k} \mathcal{E}_{rs} & \text{if undirected} \\ \bigcup_{r=1}^{k} (\mathcal{E}_{rs} \cup \mathcal{E}_{sr}) & else, \end{cases} \tag{5.2}$$

must be larger or equal to the number of vertices in group s, i.e. $\mid \mathcal{E}_s \mid \geq \mid \mathcal{V}_s \mid$. The set $\mathcal{E}_s$ of incident edges includes inter group- as well as intra group edges. The

number $\mid \mathcal{E}_s \mid$ of incident edges also follows a Poisson distribution with mean:

$$\lambda_{e_s} = \begin{cases} \sum_{r=1}^{k} \mathbb{I}_{r \neq s} \mid \mathcal{V}_r \mid\mid \mathcal{V}_s \mid (M_{rs} + M_{sr}) + \mid \mathcal{V}_s \mid^2 M_{ss} \text{ if undirected} \\ \sum_{r=1}^{k} \mathbb{I}_{r \neq s} \mid \mathcal{V}_r \mid\mid \mathcal{V}_s \mid M_{rs} + M_{ss}(\frac{1}{2} \mid \mathcal{V}_s \mid (\mid \mathcal{V}_s \mid -1) + \mid \mathcal{V}_s \mid) \text{ else.} \end{cases} \tag{5.3}$$

If $\lambda_{e_s} >> \mathcal{V}_s$ for all groups s, or equivalently $d(v) > 1 \forall v \in \mathcal{V}$ then the proposed algorithm leads to a connected graph.

Rewiring Algorithm A connected graph is obtained under the constraints given above by rewiring edges. The pseudo-code is given in Algorithm 2. An unconnected graph is likely to be generated by any of the models if the empirical graph is sparse. Then, the given condition for a connected graph can be fulfilled, but some vertices might end up with multiple edges and others with no edges. The presented algorithm targets this case and merges unconnected components by rewiring edges. The proposed algorithm performs well in practice as the results will show. The algorithm works as follows: For each unconnected component $C = (\mathcal{V}^c \subset \mathcal{V}, \mathcal{E}^c \subset \mathcal{E})$, a vertex $v \in \mathcal{V}^c$ is chosen with:

- uniform probability in case of the SBM,
- probability proportional to θ in case of the DCBM,
- probability proportional to $\theta^{out} + \theta^{in}$ in case of the DDCBM.

Then the set $\mathcal{Q} := \{w \in \mathcal{V}/\mathcal{V}^c \mid z_w = z_v \wedge d(w) > 1\}$ of vertices in the same group as vertex v, with a degree larger one and not part of C is obtained. This corresponds to lines two to six in Algorithm 2. If $\mid \mathcal{Q} \mid = 0$, i.e., the condition in line seven is not fulfilled, then all vertices in z_v are either in $\mathcal{Q}$, and/or have a degree of one. The unconnected component C is then updated to $C' = (\mathcal{V}^{c\prime}, \mathcal{E}^{c\prime})$ with $\mathcal{V}^{c\prime} := \mathcal{V}^c / \{w \in \mathcal{V}^c \mid z_w = z_v\}$ and $\mathcal{E}^{c\prime} := \{(m,n) \in \mathcal{V}^{c\prime} \times \mathcal{V}^{c\prime} \mid (m,n) \in \mathcal{E}^c\}$ by removing all vertices with the same group membership as z_v in line ten. This is necessary, since $\mid \mathcal{Q} \mid = 0$ will hold for them as well. The process of drawing a vertex and pruning C, i.e., the while loop starting in line four, is continued until either a vertex is found or $\mid \mathcal{V}^{c\prime} \mid = 0$. In the latter case C cannot be merged and G stays unconnected. If the condition line line seven evaluates to true, i.e., $\mid \mathcal{Q} \mid > 0$, then a vertex $q \in \mathcal{Q}$ is drawn from $\mathcal{Q}$ in line 15 with:

- uniform probability in case of the SBM,
- probability proportional to θ^{-1} in case of the DCBM,
- probability proportional to $(\theta^{out} + \theta^{in})^{-1}$ in case of the DDCBM.

Now the set $\mathcal{N}'_q := \{m \in \mathcal{N}_q \mid d(m) > 1\}$ of vertices adjacent to q with degree larger one is obtained. If $\mid \mathcal{N}'_q \mid > 0$, take any edge incident to q and place it on v. Thus, the connected component is merged (lines $16 - 20$). If $\mid \mathcal{N}'_q \mid = 0$ discard q and draw a new vertex from $\mathcal{Q}/\{q\}$. This is necessary because a degree of one indicates a leave. If all adjacent vertices are leaves, then moving the edge from q to v only increases C by the respective vertex. If $\mid \mathcal{Q}/\{q\} \mid = 0$, then C is pruned as before and the whole process repeated.

After one iteration over all components, the graph is again checked for connectedness and if required, the procedure is repeated until the graph is connected. Successful termination of Algorithm 2 results in a connected multi-graph.

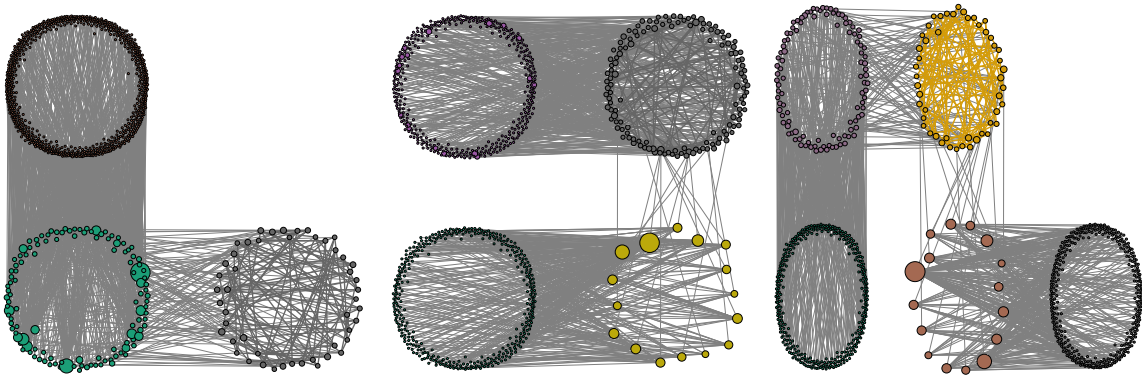
Obtaining simple Graphs If the resulting graph is required to be a simple directed or undirected graph instead, a last pass over the graph is necessary to remove possible multi- and self edges.

5.2.5 Reference Generator

Orbis [MHK+07] serves as reference generator in this chapter. Synthetic graphs generated with the block models are compared to graphs generated with Orbis. Generating synthetic graphs with Orbis follows a similar procedure as generation with any of the models introduced in this thesis. An empirical graph is used as input, characteristics are extracted, and synthetic graphs are generated based on those characteristics. In case of Orbis, the characteristic being preserved is a dK-distribution. The term dK-distribution refers to an degree distribution of arbitrary order d, i.e., $d = 0$ corresponds to the average degree. The $1K$-distribution refers to the degree distribution, and the $2K$-distribution to the joint-degree distribution. Orbis preserves either the $1K$- or $2K$-distribution by randomly rewiring edges. If $d = 2$, the rewiring of any edge must result in a graph with the same joint degree distribution [MKFV06]. It is easy to see how $d = 2$ constraints the set of possible graphs more strongly than $d = 1$. As a result, synthetic graphs will resemble the original graph more closely [MKFV06, MHK+07].

In addition, Orbis is able to scale the distributions of input topologies, and therefore generate graphs with varying number of vertices [MHK+07]. Scaling the distribution is tailored to the Internet graph, since dK-distributions of graphs arising in social sciences or biology evolve differently.

In this work, Orbis is used to generate graphs with the same $2k$-distribution as the original graphs. Evaluation of properties from synthetic graphs generated with any of the introduced probabilistic and having varying number of vertices is left for future work. Thus, the capability of Orbis to scale the input distribution is not used.

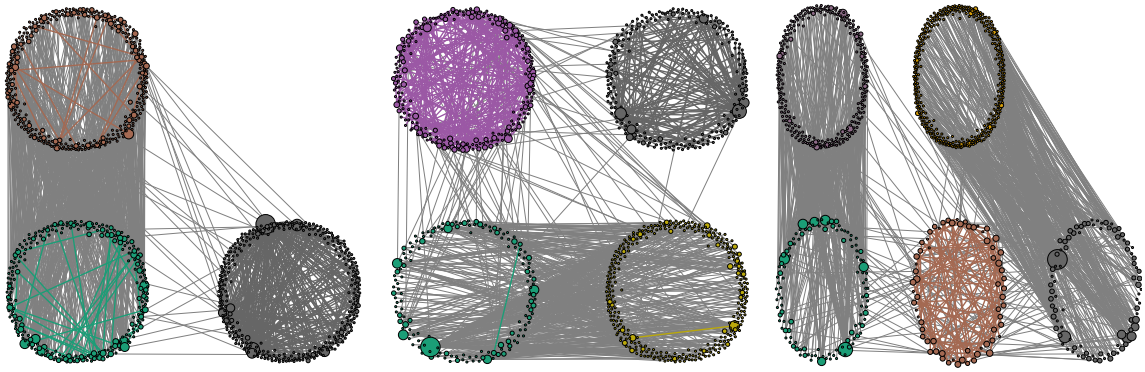


(a) SBM, $k = 3$ (b) SBM, $k = 4$ (c) SBM, $k = 5$

Figure 5.2: Structure of HOT graph inferred with the SBM for different k. The groups are numbered ascending from bottom to top and left to right starting with one.

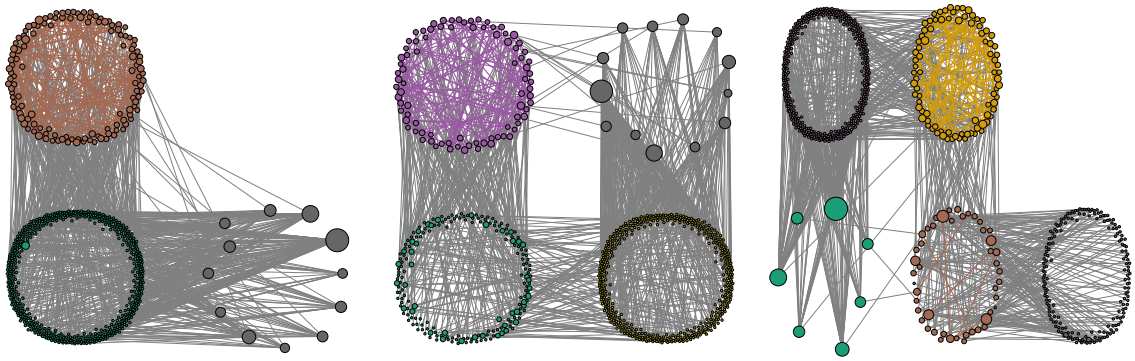
5.3 Evaluation

To obtain reliable results, 100 synthetic graphs are generated with each approach and average values and standard deviations are calculated for ten different graph metrics on the largest connected component. The remainder of this section discusses for each graph the inferred structure of the different models, differences in the resulting synthetic graphs and compares the results to graphs generated with Orbis.



(a) DCBM, $k = 3$ (b) DCBM, $k = 4$ (c) DCBM, $k = 5$

Figure 5.3: Structure of HOT graph inferred with the DCBM for different k. The groups are numbered ascending from bottom to top and left to right starting with one.



(a) Orbis $\rightarrow$ SBM, $k = 3$ (b) Orbis $\rightarrow$ SBM, $k = 4$ (c) Orbis $\rightarrow$ SBM, $k = 5$

Figure 5.4: Structure of synthetic graph generated with Orbis and inferred with the SBM for different k. The groups are numbered ascending from bottom to top and left to right starting with one.

Table 5.2: Graph Metrics for SBM, DCBM and Orbis on the HOT Graph.

	Original	DCBM	SBM	SBM+DCBM	Orbis2k
Edges	988.00	**996.65** ± 20.97	1 003.78 ± 21.96	1 012.24 ± 27.82	822.11 ± 20.57
Closeness	0.15	0.12 ± 0.02	0.12 ± 0.01	**0.14** ± 0.01	0.16 ± 0.00
Assortativity Coefficient	−0.22	−0.16 ± 0.02	−0.45 ± 0.01	−0.25 ± 0.02	**−0.24** ± 0.01
Clustering Coefficient [e^{-3}]	0.00	1.86 ± 1.41	0.80 ± 0.82	**0.79** ± 0.08	1.81 ± 0.28
Path Length	6.81	9.44 ± 2.13	8.41 ± 1.08	7.42 ± 0.87	**6.23** ± 0.08
Nodes	939.00	**939.00** ± 0.00	**939.00** ± 0.00	**939.00** ± 0.00	754.25 ± 21.25
KNN	4.00	4.37 ± 0.55	3.38 ± 0.25	**4.03** ± 0.39	4.26 ± 0.02
Degree	2.10	**2.12** ± 0.04	2.14 ± 0.05	2.16 ± 0.06	2.18 ± 0.01
Corenesse	1.16	1.10 ± 0.03	**1.17** ± 0.04	1.13 ± 0.04	1.12 ± 0.01
Neighbor Degree	19.00	11.82 ± 1.39	14.47 ± 0.50	**20.13** ± 1.08	21.53 ± 0.70

Average of respective graph feature plus/minus one standard-deviation calculated over 100 samples. Closest value is indicated in bold font.

Table 5.3: Graph Metrics for SBM, DCBM and Orbis on the LBNL Graph.

	Original	DCBM	SBM	SBM+DCBM	Orbis2k
Edges	12 329	12 356.70 ± 116.45	**12 334.71** ± 113.48	12 293.33 ± 116.0	14 724 ± 12.22
Closeness	0.19	0.16 ± 0.00	**0.17** ± 0.00	0.15 ± 0.00	0.30 ± 0.00
Assortativity Coefficient	−0.31	**−0.32** ± 0.00	−0.57 ± 0.01	−0.34 ± 0.00	−0.34 ± 0.00
Clustering Coefficient	0.10	0.07 ± 0.03	0.02 ± 0.00	**0.09** ± 0.00	**0.09** ± 0.00
Path Length	2.87	2.10 ± 0.04	**2.34** ± 0.04	3.38 ± 0.01	3.38 ± 0.01
Nodes	2581.00	**2581.00** ± 0.00	**2581.00** ± 0.00	**2581.00** ± 0.00	**2581.00** ± 0.00
KNN	62.96	47.49 ± 1.15	39.22 ± 1.20	**58.24** ± 0.32	**58.24** ± 0.32
Degree	9.55	9.58 ± 0.09	**9.56** ± 0.09	11.41 ± 0.01	11.41 ± 0.01
Coreness	6.71	**5.42** ± 0.06	5.21 ± 0.0	8.56 ± 0.01	8.56 ± 0.01
Neighbor Degree	155	**111.72** ± 2.47	65.19 ± 1.33	226.86 ± 0.75	226.86 ± 0.75

Average of respective graph feature plus/minus one standard-deviation calculated over 100 samples. Closest value is indicated in bold font.

5.3.1 Evaluation of the HOT-Graph

Since router level topologies are more engineered and less random, the introduced models are expected to infer meaningful structure.

Inference Figure 5.2 shows the structure inferred for the HOT graph by the SBM for $k = 3, 4, 5$. The groups are numbered ascending from bottom to top and from left to right starting with one. Thus, the lower left group in Figure 5.2a is referred to as Group1, the upper left as Group2 and the lower right as Group3. Figure 5.2a shows the inferred structure for $k = 3$. Here, the SBM grouped all access router into Group2. Group1 serves as aggregation layer with some very high degree vertices connecting to many vertices in the access group. Group3 provides connectivity and is connected exclusively the aggregating Group2. Figure 5.2b shows the inferred structure for $k = 4$. Compared to Figure 5.2a multiple changes are visible. Group1 in Figure 5.2a is separated into high and low degree vertices. The high degree vertices form Group3 in Figure 5.2b, while the low degree vertices form together with Group3 from Figure 5.2a Group4 in Figure 5.2b. Group2 in Figure 5.2a is separated accordingly into Group1 and Group2 in Figure 5.2b. Increasing the number of groups to $k = 5$ results in a separation of Group3 in Figure 5.2b into Group2 and Group4 in Figure 5.2c. Similar to Figure 5.2a, every group handles now again either access, aggregation or provides connectivity.

Figure 5.2c also nicely explains the relative large average shortest path distance in Table 5.2. Most of the vertices are located in Group1 and Group5 in Figure 5.2c. The minimum possible distance between any two vertices in those groups is four, and although Group2 looks strongly connected, the density is actually 0.16 only. Together with the weak connectivity between Group3 and Group4 requires additional hops. The same is true for most of the vertices in Group1. The aggregating vertices in Group2 have small degrees. Thus, communication has to go all the way up to Group4 and then down again. This results in a minimum distance of four. The same applies to vertices in Group5. Since the aggregating vertices in Group3 have large degrees, the effect is not as strong since the distance between many vertices is then only two.

Due to the clear topological roles of the different groups, the impact of changing parameters of the model can easily be assessed. For example, increasing the parameter for the intra-group connectivity of Group4 in Figure 5.2c results in a tighter mesh. This will increase the clustering coefficient and shorten the average shortest path length. Introducing inter-group communication for Group3 would result in a form of rich-club and greatly reduce distances for all vertices in Group5.

In conclusion, the SBM is able to infer meaningful structure with respect to topological roles of vertices for different values of k. The clearest separation is found for $k = 5$, for which the MDL obtained its minimum. The resulting clear

interpretation of the model's parameters allow control over structural properties. The impact of changing any parameter can easily be anticipated.

Comparing Figure 5.2b and Figure 5.2c show that for $k = 4$ essential properties are captured. Of course, the separation found with $k = 5$ is much clearer. Using the structure inferred for $k = 4$ allows more variation of the resulting topologies without influencing the overall structure too much. As stated before, variability is also an criterion that should not be dismissed easily. Using $k = 3$ would not preserve the topological characteristics of the graph. Due to the SBM being a mixture of $G(n, p)$ models, the degrees in Group1 in Figure 5.2a would not be retained. As stated earlier, the degree heterogeneity of that group heavily impacts connectivity properties of the graph.

The separation found by the DCBM for $k = 3, 4$, depicted in Figure 5.3a and Figure 5.3b, does not result in distinctive groups. Only the separation for $k = 5$ in Figure 5.3c resembles the respective separation found by the SBM. However, many vertices being in separate groups for the SBM are in the same group for the DCBM. Actually, the separation found by the SBM yields a larger likelihood when used in conjunction with the DCBM. Let z_{SBM} and z_{DCBM} be the inferred group membership for SBM and DCBM respectively, a DCBM using z_{SBM} is more likely to have generated the HOT graph than a DCBM with z_{DCBM}. The reason lies in the objective function and the algorithm used to find the parameter z. In case of the DCBM, it gets caught in a local optima. The SBM is thus superior to the DCBM with respect to inferring the structure of the HOT graph.

In summary, the results of this section so far hint at the SBM to be well suited to infer meaningful groups with respect to structural roles of vertices for router-level graphs. The evaluation of additional graphs is needed to verify this and is left for future work.

Generation Figure 5.5 shows the CCDF of the degree distribution for the HOT graph and one synthetic graph per model. The x-Axis depicts the (vertex) degree K, and the y-axis the probability of $K \geq k$ for some $k \in \mathbb{N}$. The SBM has difficulties to generate the heavy tail of the degree distribution. As mentioned in Section 2.4, vertices are stochastically equivalent given z. Nodes in the same group are thus equally likely to receive an edge, and therefore have the same expected degree. This flattens out the heterogeneous degrees of Group1 in Figure 5.2b and leads to the step visible in Figure 5.5.

The DCBM fits the degree distribution better than the SBM, despite failing to capture the topological roles of vertices. This is expected, as the DCBM models the expected degrees of the vertices and is thus able to better reproduce heavy tailed degree distributions. Modeling of the expected degrees allows the DCBM additionally to compensate the deficiencies in identifying topological roles.

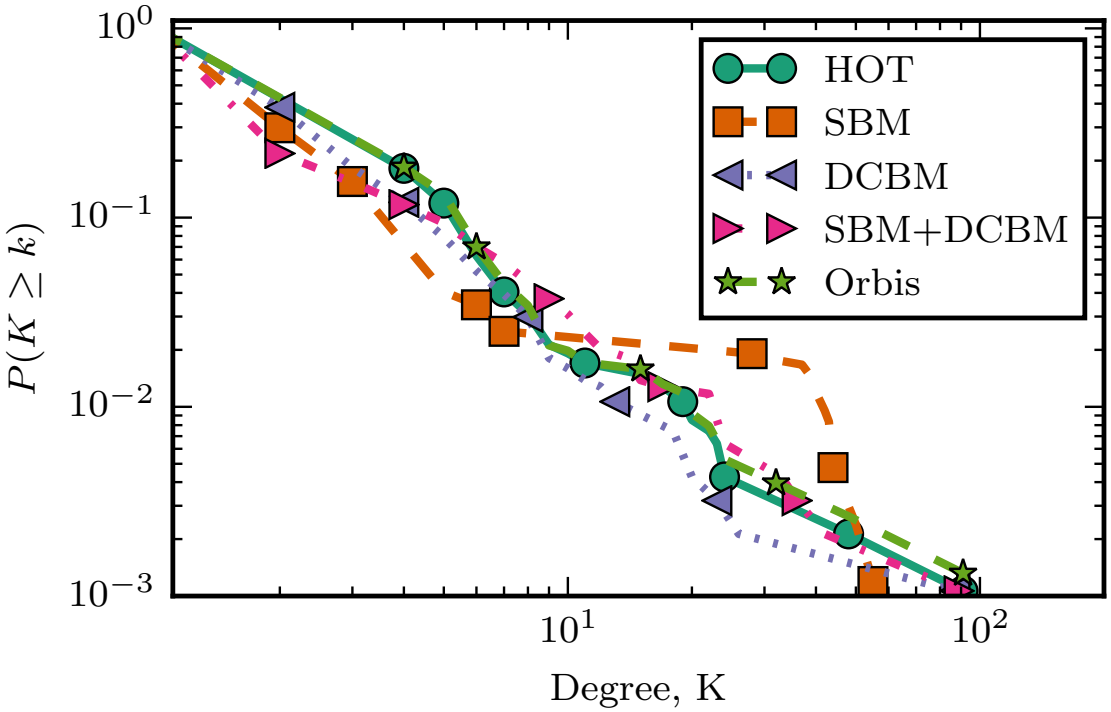


Figure 5.5: CCDF of the degree distributions of HOT- and synthetic graphs.

The DCBM has problems identifying the underlying structure, while the SBM fails to model the heavy tail of the degree distribution, having an impact on other graph attributes. The SBM+DCBM combines the respective strengths, resulting in an even tighter fit to the original CCDF. Also, most of the graph features are closer to the original values (Table 5.2). This highlights the importance of incorporating structural and degree information.

In summary, the SBM+DCBM approach is well suited for this particular router level graph. The SBM is able to group vertices with respect to topological roles and preserve this structure during generation. Heterogeneous degrees, which might be present in some groups, are also preserved.

Comparison to Orbis Orbis fits the degree distribution best (Figure 5.5). As Orbis generates graphs with the same joint degree distribution, the degree distribution itself is also preserved. Despite the almost perfect match, the synthetic graphs generated with the combined approach still outperform Orbis in terms of graph features in Table 5.2. The stronger deviation from the original graph attributes reflects the structural changes visible in Figure 5.4. The depicted structure is inferred using the SBM with $k = 3, 4, 5$ on the largest connected component of a synthetic graph generated by Orbis. The values for k resulted in good results previously. If Orbis preserves the topological structure, then the inferred groups should be the same as before. Indeed, the inferred structures are similar to the original ones in Figure 5.2. However, the high degree vertices in Figure 5.4a are

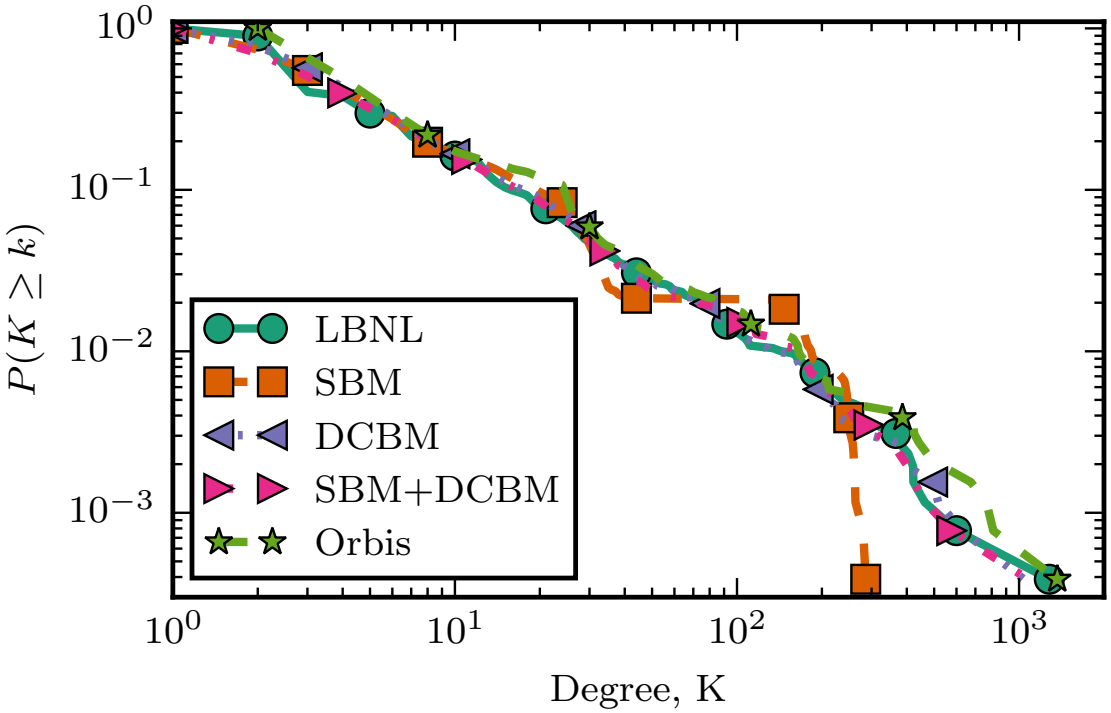


Figure 5.6: CCDF of the degree distributions of LBNL- and synthetic graphs.

located in their own group. The low degree vertices from Group1 in Figure 5.2a seem to be merged with the group providing connectivity for the structure depicted in Figure 5.4a. Aside from those differences, the structures look very similar. For $k = 4$ more structural differences become apparent. Figure 5.4b shows a strong connectivity between the groups containing leaves. Similar, for $k = 5$ structural differences are observable in Figure 5.4c. Here, Group2 contains aggregating vertices, as well as vertices providing connectivity. Also, Group3 contains intra-group edges, and most of the leaves are located in Group2. The structural changes are introduced as Orbis randomly rewires vertices and is constraint only by the given joint degree distribution. SBM and DCBM also randomly rewire vertices, however with respect to the inferred parameters, which encode the structure of the graph. Thus SBM and SBM+DCBM generate graphs with the structure shown in Figure 5.2. Depending on the use-case, graphs generated with the SBM might thus be preferable, despite a stronger deviation of the values from graph features.

5.3.2 Evaluation of the LBNL Graph

This section gives the result for inference and generation of synthetic graphs for the LBNL graph found and produced by the DCBM and DSBM, as well as a comparison to Orbis.

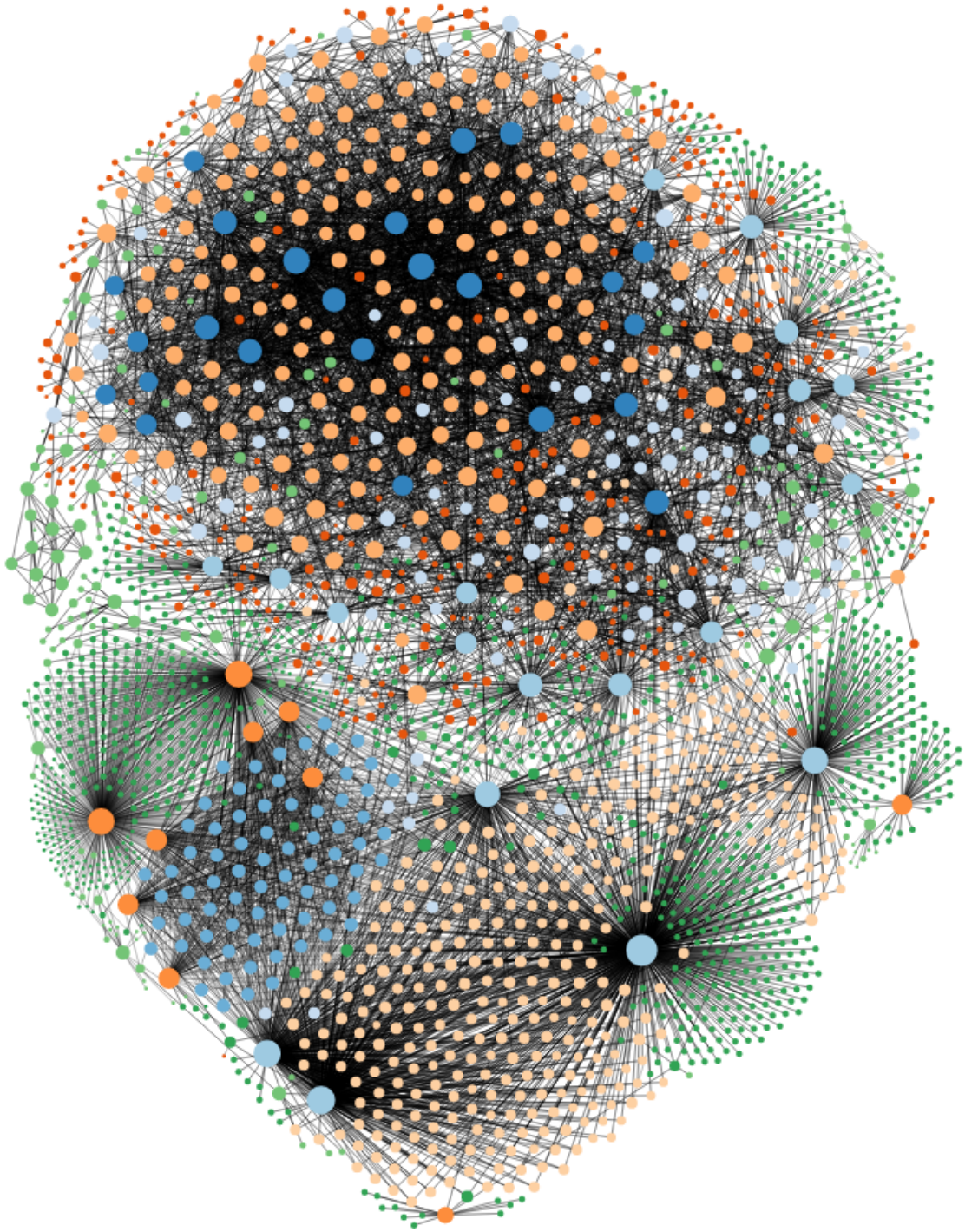


Figure 5.7: Force directed visualization of the LBNL graph with groups inferred with the DSBM indicated by different colors, The size of the vertices is proportional to their degree.

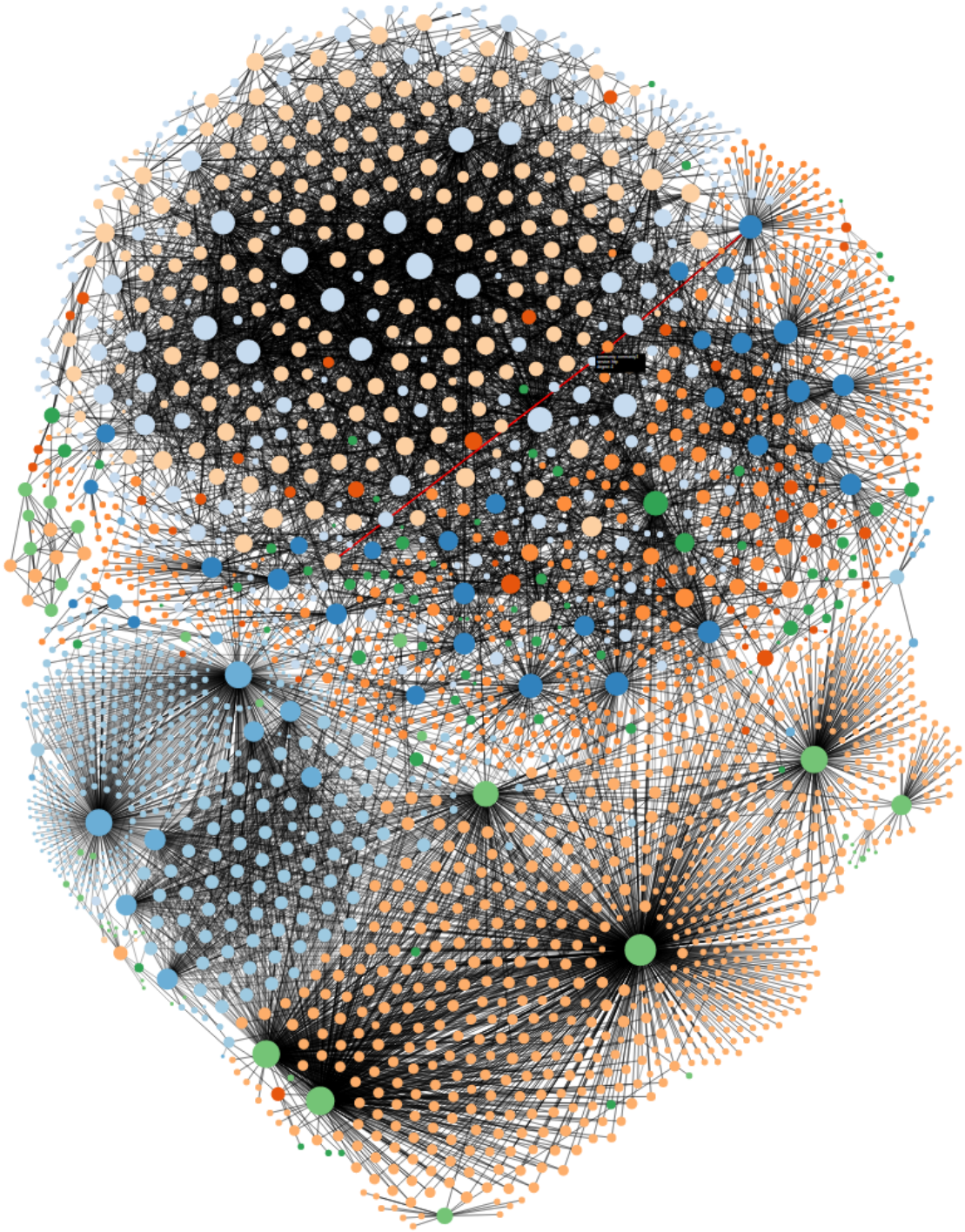


Figure 5.8: Force directed visualization of the LBNL graph with groups inferred with the DDCBM indicated by different colors, The size of the vertices is proportional to their degree.

Inference The LBNL Graph has a more complex structure than the HOT graph. Still, 10 groups are adequate for the DDCBM to describe the structure of the graph according to the MDL principle. The large number of groups prohibits a visualization as in Figure 5.2, as well as a detailed description of all the groups. Therefore, Figure 5.7 and Figure 5.8 show a force directed layout of the LBNL graph. The force directed layout results in roughly the same position for vertices of the same graph every time the graph is visualized and allows thus a comparison. The color of the vertices indicate group membership, the size of the vertices is proportional to their logarithmic degree.

Figure 5.8 illustrates why ten group are sufficient for the DDCBM to represent the graph. Consider for this the upper left part of the graph. This part contains vertices with very large degrees around 1 000 and vertices with low degree at the edge and in the middle. Both, the vertices with high degree and low degree are placed in the same group, indicated with the light blue color. This is the pattern identified in Section 4.2.2, Figure 4.4. The model identifies a bipartite like structure due to its ability to accommodate high and low degree vertices in one group, i.e., vertices in the light blue group mostly connect to vertices of the light orange group. The same is true for the deep blue and orange group in the right part of the upper region of Figure 5.8. By exploiting this structure, the DDCBM is able to compress the structure of the graph reasonably well with only 10 groups. Assigning specific structural roles to the groups is, similar as in the previous section, not possible for the split fount by the DDCBM.

The DSBM in contrast is not able to exploit this structure and thus requires more groups as indicated with the minimum for the MDL for $k = 21$. However, for $k = 10$ reasonable groups with respect to structural roles are identified. Leaves are accommodated in different groups than more connected vertices, as Figure 5.7 shows. The SBM also separates between the star structure and the more regular structure in the left part of the lower section of the graph in Figure 5.7, i.e. the green, blue and orange groups respectively.

Contrary to earlier, the split found by the DSBM does not lead to a larger likelihood when used with the DDCBM. Thus, under the DDCBM, the split of the DSBM is less likely to generate the observed graph. This is expected, since the DSBM cannot use the same structural properties, as it cannot accommodate vertices with strongly varying degrees in one group.

In conclusion, the DSBM does yield a separation, in which structural properties can clearly be assigned to different groups. If the goal is structural analysis, the SBM is thus better suited to the task than the DDCBM. However, if the inferred structure is of little interest, then the DDCBM provides a more concise representation. The next paragraph investigates how the different separations influence the synthetic graphs.

Generation The DSBM cannot generate the observed heavy tail as the CCDF in Figure 5.6 shows. The DDCBM and the SBM+DCBM approach have an almost perfect fit. Surprisingly, the DSBM replicates graph features in Table 5.3 better than DDCBM and SBM+DCBM. However, the DSBM fails at reproducing the clustering coefficient, which is an important metric [MKF+06]. Synthetic graphs generated with the DSBM have an average clustering coefficient of only 0.02, which es still larger than the expected $2.24e^{-3}$ based on a random graph, but also significantly less than the observed 0.1. In contrast, for the DDCBM the value of 0.1 for the clustering coefficient is contained within one standard deviation, and the SBM+DCBM consistently achieves a value of 0.9. Therefore, graphs generated with the DCBM or SBM+DCBM approach are better, since they do not have such strong deviations from other graph properties.

Comparison with Orbis DSBM and DDCBM reproduce the structure they inferred previously. Interestingly, Orbis also has slight problems with the heavy tail of the LBNL graph (Figure 5.6) and is outperformed by the DDCBM and DSBM+DDCBM. This may be due to Orbis being able to generate undirected graphs only, and thus not being able to handle asymmetric communication, which the DSBM and DDCBM can capture. This also explains the relative large number of edges and average neighbor degree for synthetic graphs generated by Orbis in Table 5.3.

5.4 Summary

This chapter shows that probabilistic models, namely the Stochastic-Block-Model (SBM) and the Degree-Corrected-Block-Model (DCBM), as well as their directed variants, the Directed Degree Corrected Block Model (DCBM) and Directed Stochastic Block Model (DSBM), greatly assist in the analysis of communication networks and their synthetic generation. The models are able to generate synthetic graphs from varying network domains. The inferred structure of the SBM and DSBM results in a compressed, yet well interpretable model, capturing essential structural roles for vertices, as shown for a router-level graph, and preserving those roles in synthetically generated graphs.

With respect to inference of meaningful structure, the SBM and DSBM have proven themselves superior to the DCBM and DDCBM in the set-up of this chapter. However, the additional capabilities of the DCBM and DDCBM to model individual vertex degrees are important to generate graphs with observed heavy tailed degree distributions, as shown for an IP-to-IP communication graph. Additionally, the DDCBM and DCBM may provide very concise but structurally not very well interpretable representations of IP-to-IP graphs. By joining the infer-

ence capabilities of SBM and DSBM, and the generative capabilities of DCBM and DDCBM, this chapter shows that the models outperform the Orbis network generator.

This chapter also shows how the free parameter k can be chosen if no prior knowledge exists. An interesting future research direction would be the generation of graphs with varying number of vertices using more complex models, as well as the generation of dynamic graphs [Pei14, ZYM12, HSM14, XI13].

Chapter 6

Anomaly Detection

Based on the insights from Chapter 4 and Chapter 5, the Directed Stochastic Block Model (DSBM) is used in this chapter for an proof of concept for detecting behavioral anomalies in an unsupervised fashion. More precisely, hosts taking part in a botnet are identified in a university network.

This chapter is organized as follows, Section 6.1 gives the problem statement. Section 6.2 introduces the proposed method to detect behavioral anomalies, and Section 6.3 discusses the results. Finally, Section 6.4 closes this chapter by giving a short summary and outlook.

6.1 Problem Statement

Detecting behavioral anomalies, i.e., hosts participating in a botnet, can be cast as detection of point or collective anomalies [Sta12, XWG11], or contextual anomalies independent of time [Sta12]. The latter approach is used in this thesis. The challenge is then the provision of a context (see Section 2.3.2) in which the behavior of hosts is evaluated, as well as the definition of normal behavior [DHB+10]. Those problems are tackled in this chapter using the DSBM and a graph representation of the IP-to-IP communication of hosts. The context of a host, i.e., the other hosts it is compared to, is then given by members of the same group. Behavior of each host can be quantified with the joint probability of outgoing edges under the DSBM of each host.

Additional challenges arise due to the dynamic nature of the communication. Over time, hosts accumulate edges as they consume different services. This can hide the anomalous behavior, as the number of anomalous interactions can be small compared to the overall number of interactions for one host. If the impact of anomalous interactions is large, then the host's group membership inferred with the DSBM might change [MM15]. Consequently, the context changes as well, po-

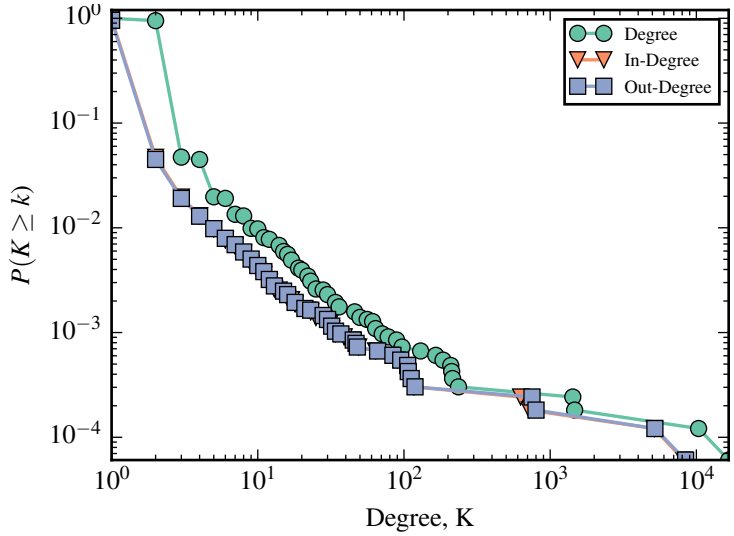


Figure 6.1: CCDF of graph extracted for the first time window.

tentially hiding anomalous behavior. To counteract those effects, communication is sliced into fixed sized windows. From each window, a graph is extracted and model parameters of the DSBM re-estimated. Group membership is assumed to be fixed, i.e., group membership is estimated once for each host and then used in subsequent time steps.

An entirely different problem is the availability of ground truth. While labels are not used during parameter inference of the DSBM and identification of potential anomalous hosts (i.e., the proposed approach is unsupervised), they are still important to *verify* whether identified hosts are indeed anomalous. The data set used in this chapter provides ground truth for ten hosts. Results reported in Section 6.3 are therefore restricted to those ten hosts.

6.2 Material and Methods

This section describes the used data and methodology for a proof of concept on using the DSBM for botnet detection. Section 6.2.1 presents the used data set and data preparation tasks. Section 6.2.2 details the process of fitting the DSBM, and 6.2.3 describes metrics used to label hosts as normal or anomalous.

6.2.1 The Data Set

This section firstly describes the raw-data, how it is processed and finally the processed data used to perform anomaly detection.

The data set consists of a $9.8\,GB$ PCAP file containing traffic of the CVUT University in Prague, Czech[1], and ground truth for ten hosts. The rest of this chapter refers to those ten hosts as H0 to H9. The data set is released as part of the IPS Stratosphere project, founded at the CVUT University, Prague and funded by the NLnet Foundation[2]. The data is publicly available at [Gar].

The PCAP files contains approximately $4.6\,h$ hours of traffic, from which two hours are background and normal traffic. Normal traffic refers to traffic of hosts that are known to be not infected by any malware. Background traffic refers to traffic of hosts not under control of the authors. Thus, no statement about the *normality* of their traffic can be made. After the initial two hours, a total of 10 hosts is infected with the Neris malware over the course of 40 minutes in an controlled environment by the authors of the data set. After infection, the capture continues for another two hours. The trace contains a total of 346 000 different IP addresses.

The PCAP file is sliced into time windows of $5\,min$ length. The value of $5\,min$ is chosen after evaluating the arrival of hosts for different window lengths. For smaller values, the number of vertices in neighboring time windows greatly fluctuates. This has the potential to interfere with group estimation. For a length of $5\,min$ the values are more stable. Evaluating the impact of window size is left for future work.

This results in 55 different graphs with approximately 20 800 vertices and 42 600 edges on average. Infection of the first machines takes place in window 23 and continues until window 32. Detailed values for all hosts are given in Table 6.1. Apart from the time window in which hosts are infected by the authors of the data set, the table provides information about the time window each host appeared for the very first time, the time window each host appeared the next time after infection, and in which time window each host is labeled as anomalous by the DSBM. The table also provides the number of false alarms, i.e., the number of times the DSBM labeled a host as anomalous prior to infection, and the corresponding false positive rate. Approximately 15 000 new IP-addresses appear between successive time steps on average. From those, about 6 000 have never appeared before. This indicates a high fluctuation in IP-addresses. Indeed, approximately 90 % of all IP-addresses appear only up to five times during the $4.6\,h$ capture, and most of the hosts are clients. This is illustrated in Figure 6.1. Figure 6.1 depicts the CCDF for the degree, in-degree and out-degree distribution of the graph extracted from the first time window. Figure 6.1 shows that almost 100 % of the vertices have a degree of only two and most of the communication is bidirectional, indicated by the overlap of in- and out-degree. Consequently, almost 100 % of the vertices communicate

[1]Last accessed: 02.03.2017

[2]See https://stratosphereips.org/category/about.html

with only one single endpoint. The high density of markers between a degree of ten and 100 indicates strong variability in this area. Figure 6.1 additionally shows the existence of few vertices with a degree larger than 10 000. The CCDF for other time steps closely resembles the one in Figure 6.1.

6.2.2 Estimating SBM over Time

As Chapter 4 shows, the DSBM successfully separates between hosts and server. The assumption of constant group assignment is thus equivalent to assuming that clients stay clients and server stay server over time. The analysis assumes that a vertex keeps the first assigned group label for all subsequent time steps. Thus, at the first time step, group memberships for all vertices are estimated. In subsequent time steps, group membership for only 6 000 new vertices, appearing on average, needs to be estimated.

This approach underlies the strong assumption, that hosts are assigned to the "correct" group on the first try, i.e., high confidence is placed on the estimated parameters. Fluctuations that might lead to an erroneous assignment are not taken into account. The same is true for the uncertainty of inferred group membership in general. Evaluation of alternative approaches considering the uncertainty in parameters are left for future work.

The hosts going to be infected appear according to Table 6.1 in the first few time windows. Table 6.1 also shows that hosts are infected many time windows after their first appearance. H0 to H9 are thus assigned to a group based on their normal behavior. The assigned group then provides the context against which the behavior of the hosts is evaluate when infected.

In order to obtain a good number of communities, the DSBM is fit to the graph extracted from the first time window with multiple values for k, i.e., the parameter controlling the number of groups. Model selection is then performed using MDL as in previous chapters. The minimum value for the MDL is obtained for $k = 6$. This value is kept constant over time.

6.2.3 Identifying anomalies

At each time step, the joint probability for outgoing edges of each vertex is calculated, given the parameters $\hat{M}^t, \hat{z}^t$ of the DSBM, i.e., the MLE of the stochastic block matrix and group membership vector. *Outgoing edges* includes observed as well as unobserved edges. The term *unobserved edges* refers to entries with value zero in the adjacency matrix. [3]. Incoming edges are ignored, as they are not

[3]A subtle but important differences exists in interpreting unobserved edges. Some edges might simply be impossible, i.e., the interactions they correspond to cannot happen. In contrast,

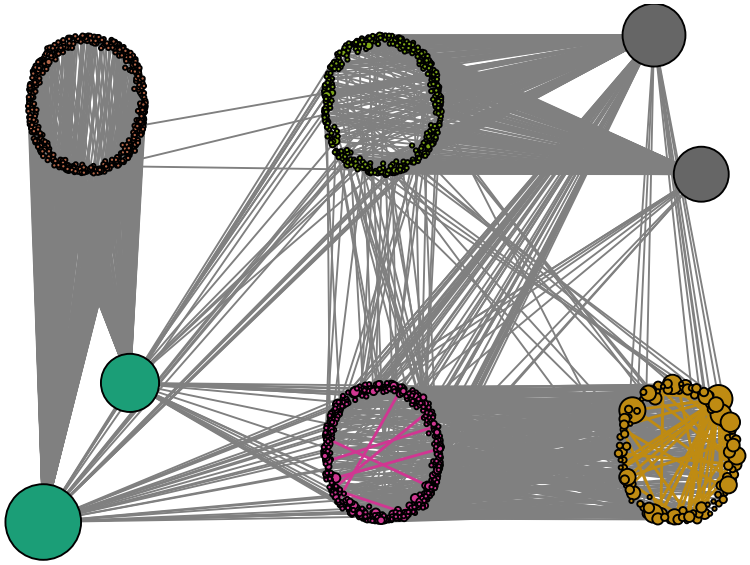


Figure 6.2: Group structure of the graph extracted from the first time window. The groups are labeled as follows: lower left is G2, upper left is G3, lower central is G5, upper central is G1, lower right is G4 and upper right is G6.

caused by the vertex in question and thus contribute no information regarding its behavior. The likelihood, i.e., joint probability, of outgoing edges of a vertex v at time t given the estimated model parameters is calculated as:

$$l_v^t = \prod_{w \in \mathcal{V}^t} p(A_{vw}^t \mid \hat{z}^t, \hat{M}^t), \tag{6.1}$$

where the CI of edges given the model paramters is used. To avoid underflow, the logarithm of l_v^t is taken. Numerical issues are inevitable due to the large number of multiplications performed in Equation 6.1. For each vertex, $\mid \mathcal{V}^t \mid$ probability values, i.e., 20 800 on average for the used data set, are multiplied in order to obtain the likelihood. As probability values are generally smaller one, underflow is inevitable. Taking the logarithm has the nice side effect of simplifying calculations.

other edges, i.e., interactions, are possible but did simply not happen. The two different kinds of unobserved edges should in general be treated differently [AJC15]. This, as so much else, is left for future work.

The log likelihood $\log l_v^t$ of each vertex v at time t is given as:

$$\begin{aligned}
\log l_v^t &= \sum_{w \in \mathcal{V}^t} \log p(A_{vw}^t \mid \hat{z}^t, \hat{M}^t) \\
&= \sum_{w \in \mathcal{V}^t} \log \left(\frac{\left(\hat{M}_{\hat{z}_v^t \hat{z}_w^t}^t \right)^{A_{vw}^t}}{A_{uv}^t!} \exp(\hat{M}_{\hat{z}_v^t \hat{z}_w^t}^t) \right) \\
&= \sum_{w \in \mathcal{V}^t} \left(A_{vw}^t \log \hat{M}_{\hat{z}_v^t \hat{z}_w^t}^t + \hat{M}_{\hat{z}_v^t \hat{z}_w^t}^t \right) \\
&= \sum_{r=1}^{k} \left(e_{v,r}^t \log \hat{M}_{\hat{z}_v^t r}^t + \mid \mathcal{V}_r^t \mid \hat{M}_{\hat{z}_v^t r}^t \right).
\end{aligned} \tag{6.2}$$

Where the fact of A^t representing a simple directed graph is used. The set $\mathcal{V}_r^t$ contains all vertices assigned to group r at time t. The number $e_{v,r}^t \in \mathbb{N}$ represents the number of outgoing edges pointing from vertex v to other vertices in group r at time t. The log likelihood for all vertices can be computed in time $\mathcal{O}(\mid \mathcal{E}^t \mid + \mid \mathcal{V}^t \mid + k)$ by pre-computing the sizes of each group. Based on the log likelihood, an anomaly score s_v^t at time t is calculated for each vertex v:

$$s_v^t := \mid \operatorname{med}_{\hat{z}_v^t} - \log l_v^t \mid \tag{6.3}$$

where $\operatorname{med}_{\hat{z}_v^t}$ is the median of log likelihood values from vertices of a single group. The median is used instead of the mean, since the median is more robust to outliers and thus a more reliable estimator of location [LLK+13]. A large anomaly score corresponds to a large deviation from the median log-likelihood and thus indicates strong behavioral differences. A small anomaly score accordingly imply normal behavior.

The anomaly scores can be used to obtain a ranking per group or across all vertices. For this, the *unique* anomaly scores are sorted descending (most anomalous top, least anomalous bottom). The rank of each vertex is then the index of its anomaly score in the sorted list. The ranking assigns meaning to the anomaly scores, i.e., given the anomaly score for a vertex, the rank can be retrieved, which immediately tells whether the likelihood score corresponds to an anomalous or normal behavior.

Due to fluctuations in connectivity and number of hosts, comparing ranks across time steps is difficult. A rank of 1 000 in one time step is not necessarily the same in other time steps. To allow comparisons across time steps the normalized rank is introduced. Rank values are normalized with the minimum (i.e., one) and maximum (time dependent) possible rank in each time window. A normalized rank of zero implies the highest possible anomaly score, i.e., the respective host

is most anomalous in the current time window. A normalized rank value of one implies the least possible rank. The respective host is thus the least anomalous host in the current time step.

In order to identify anomalous vertices, the $99th$-percentile of the anomaly scores of vertices in one group is used as cut-off value. All vertices with a larger anomaly score, i.e., with a log likelihood whose distance to the median is larger, are thus assumed to be anomalous. The $99th$-percentile is used to be certain about the reported anomalies. After all, 99 % of the other values, i.e., distances to the median, are smaller. A detailed analysis of thresholds is left for future work. The set of anomalous vertices $\mathcal{V}_A^t$ is then given as:

$$\mathcal{V}_A^t := \{v \in \mathcal{V}^t \mid s_v^t > s_{\hat{z}_v^t, p_{99}}^t\}, \tag{6.4}$$

where $s_{\hat{z}_v^t, p_{99}}^t$ is the $99th$-percentile of anomaly score values of vertices in group $\hat{z}_v^t$.

A frequent technique to identify anomalies based on robust estimators of scale [LLK+13], such as the MAD, is not applicable here. The MAD and alternative measures of scale presented in [RC93] frequently evaluate to zero. This results in a high false positive rate, since small deviations are already labeled as anomalies. This is caused by the topology of the IP-to-IP communication graphs of each time step and the representation as a simple directed graph. Figure 6.2 illustrates this. The figure shows the inferred structure of the graph extracted at the first time window. The following concentrates on groups G4 (the upper left group) and G1 (lower left group) only. Group G4 connects almost exclusively to group G1. Since communication is represented as simple directed graph, values in the adjacency matrix can only be zero or one. Considering only the vertices in G1 that communicate with G4 (which are almost all). The rows of each of those vertices in the adjacency matrix (rows correspond to outgoing edges) can take on only four different patterns (due to the binary nature of the adjacency matrix):

1. all entries are zero,
2. entry corresponding to the larger vertex in group G1 is one,
3. entry corresponding to the smaller vertex in group G1 is one,
4. both entries for the vertices in group G1 are one.

Consequently, the log likelihood score $\log l_v^t$ of each of those vertices can take on only three different values, as Equation 6.2 evaluates to the same value for pattern two and three. According to Figure 6.1, almost all vertices communicate with only a single other vertex. It can thus be assumed that most of the vertices in G1 have either pattern two or three. Those vertices will all have the same log likelihood $\log l_v^t$. This value will then be the median, resulting in an anomaly score of zero for almost all vertices. This then causes the MAD to be zero as well.

	H0	H1	H2	H3	H4	H5	H6	H7	H8	H9
First Occurrence	3	2	5	6	6	6	6	6	6	6
Infection by authors	23	28	29	30	30	31	31	31	32	32
Next Appearance	29	32	32	32	32	32	32	33	32	32
Detection (DSBM)	29	33	34	35	36	36	37	37	37	37
False Alarms	0	0	0	0	0	0	0	0	1	1
FP-Rate	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.10	0.13

Table 6.1: Row *First Occurrence* gives the time window the respective host appeared for the first time in the capture. Row *infection* gives the ime windows in which each host host is infected with Neris. Row *Next Appearance* gives the first window the host appears in after infection. This is the first possible detection point. Row *Detection* gives the time window in which the first alarm is raised. Row *False Alarms* gives the number of alarms, i.e. raised alarms prior to infection. Row *FP-Rate* gives the false positive rate corresponding to the number of false alarms.

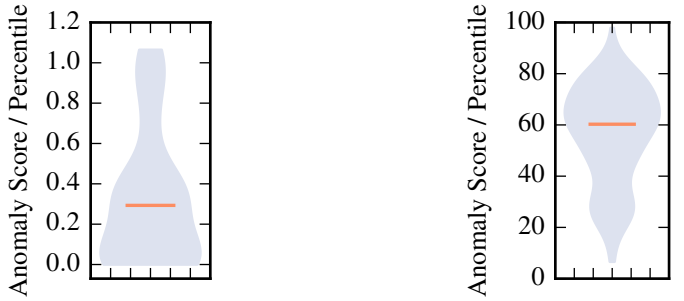
6.3 Evaluation

This section discusses the results obtained with the approach outlined in the previous sections. Section 6.3.1 describes the behavior of the ten hosts with available ground truth data prior to infection. Section 6.3.2 analyzes when the infected hosts are detected and the influence on their behavior. Section 6.3.3 finally relates the results obtained in the two previous sections to all other vertices in the graphs.

6.3.1 Prior Infection

Most hosts appear for the first time in window 6, i.e., after 30 *min* have elapsed. Exact times are given in Table 6.1. With exception of host H0, all hosts are assigned to the same group. Host H0 is assigned to group G5 and the others to group G1. Figure 6.2 shows the inferred group structure on the first time step. Subsequent time steps look similar. The groups are labeled as follows: lower left is G2, upper left is G3, lower central is G5, upper central is G1, lower right is G4 and upper right is G6. The Figure does not display all vertices and edges. For each group, a maximum of 1 000 vertices is displayed. From the edges incident to this subset of vertices, 75 % are displayed. Vertices and edges are sampled uniformly. Figure 6.2 shows, that groups G1 and G5, to which hosts H0 to H9 are assigned, correspond to client groups.

Prior infection, a total of 2 false alarms occurred, one for H8 and H9. Prior



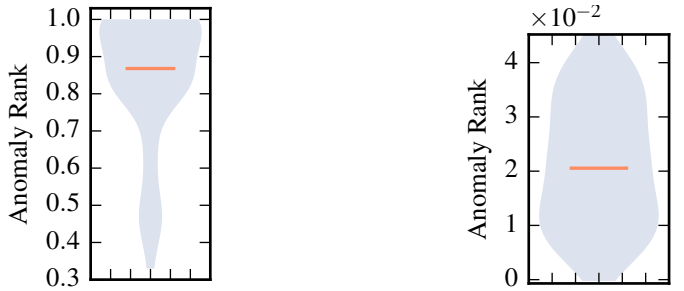
(a) Prior Infection (b) After Detection

Figure 6.3: Anomaly scores divided by the respective 99*th*-percentile. Values in Figure 6.3a are obtained from time windows prior infection. Values in Figure 6.3b are obtained from time windows after an alarm for a host is given. The blue area represents the distribution of values and the orange bar the median. Note the different scale on the y-Axis.

to infection, hosts H0 to H9 appeared in a total of 80 time windows. Two false alarms thus correspond to a false positive rate of 0.0250. For the individual hosts, the false positive rate is 0.06 for H8 and 0.10 for H9. Overall, the anomaly score for the machines is rather low prior to infection. This is illustrated in Figure 6.3a, showing a violin plot. The y-Axis represents anomaly scores divided by the value of the 99*th*-percentile. The y-Axis ranges thus from zero (minimal possible anomaly score) to infinity. A value of one indicates an anomaly score equal to the 99*th*-percentile. The shaded area in Figure 6.3a indicates the distribution of the values. The upper and lower end of the shaded area indicate the maximum and minimum value. The vertical bar represents the median. Figure 6.3a shows that most of the values are well below one, with a median of about 0.5. Thus, most of the anomaly scores are well below the value of the 99*th*-percentile, and hosts H0 to H9 confidently labeled as normal.

6.3.2 After Infection

Table 6.1 also gives the time windows of first appearance *after* infection in row *Next Appearance*, and the time window in which each host was labeled anomalous for the first time. The time window each host appears in *after* it was infected, is the first possible window in which anomalous behavior can be identified. The time between infection and next appearance varies. Host H5 and H6 appear directly in the window following infection, while H0 does not appear until six windows after infection. One would expect the time of detection to reflect this variability. Instead, the first alarm is raised for all hosts in the 5*th* or 6*th* window after



(a) Prior Infection (b) After Detection

Figure 6.4: Normalized rank of Hosts prior Infection and after Detection. At each time step, the ranks are normalized with the minimum and maximum rank possible. A normalized rank of one then indicates the highest possible (i.e. least anomalous) rank, and a normalized rank one indicates the smallest possible (i.e. most anomalous) rank. Values in Figure 6.4a are obtained from time windows prior infection. Values in Figure 6.4b are obtained from time windows after an alarm for a host is given. The blue area represents the distribution of values and the orange bar the median. Note the different scale on the y-Axis.

infection. The first host raising an alarm is H0 in window 29. The last two hosts being detected are hosts H6 to H9 in window 37. It is not clear what causes this behavior. One possible explanation could be that the malware has a sort of incubation phase in which it is inactive or listens for external commands. If this is the case, then the long delay is a general drawback of anomaly based malware detection [FSR09].

After a host is detected for the first time, an alarm is raised in all subsequent time windows. The alarm is confident, as Figure 6.3b illustrates. The anomaly score of hosts H0 to H9 is always multiple times larger than the value of the 99 % percentile, with a median value that is 60 times larger. The minimum anomaly score after detection is still 6.5 times larger than the respective percentile. Thus, the threshold could be increased, eliminating the two false positives and still identify hosts infected with the Neris malware.

In summary, the behavior of the infected hosts deviates strongly from their previous behavior, and the behavior of other hosts in their respective groups.

6.3.3 Global Scale

Comparing the anomaly score of the known infected hosts with the scores of all other vertices reveals that the change in behavior caused by the Neris malware is strong on a global scale as well. Figure 6.4 shows a violin plot of the normalized

rank assigned to hosts H0 to H9 based on their anomaly score (to obtain the ranking, anomaly scores of all hosts in respective time steps were considered). The scale of the y-Axis in Figure 6.4a and Figure 6.4b already illustrates a confident placement among the least anomalous and most anomalous hosts prior infection and after detection respectively. Figure 6.4a shows the violin plot for the normalized rank values obtained prior to infection. Most of the normalized rank values are larger than 0.7, with a median of approximately 0.85. The behavior of H0 to H9 is thus also normal compared to all other hosts in respective time steps.

Figure 6.4b shows the violin plot for the normalized rank values obtained for hosts H0 to H9 after detection. The maximum value, indicating less anomalous behavior, is smaller than 0.05, which is a tenth of the values prior to infection. The median value is 0.02 and a minimum of zero is obtained more than once. Thus, the behavior of some hosts from H0 to H9 was the most anomalous across all hosts in some time steps. The values imply that the behavior of hosts infected with Neris strongly deviates not only from their own previous behavior or the behavior of similar hosts, but also compared to the behavior of all hosts occurring in a time window.

6.4 Summary

This chapter presented an approach based on the DSBM which successfully identified hosts infected with the Neris malware. The presented approach relies on IP source and destination addresses only, and is thus robust in the presence of encryption and obfuscation. The proposed approach has an overall false positive rate of 2.5 % and a true positive rate of 100 %, once the malware becomes active.

An open question is how the proposed method compares to other methods of botnet detection [GZC14, FSR09], especially regarding the rather long time of over 20 min that passed between infection of the hosts and the first raised alarm. Another avenue for future work is the evaluation against other types of malwares such as worms. Also open remains the evaluation of additional techniques to label hosts as normal or anomalous. The goal here is decreasing the false positive rate. For this, other graph representations such as directed multi-graphs, or other probabilistic models, which are able to handle edge weights [AJC15] or additional vertex meta-date [NC15], can be investigated as well.

Chapter 7

Conclusion and Outlook

Analysis, synthetic generation, and anomaly detection are of importance for network operators. Besides all topics are intertwined by nature. Existing research suffers from either domain dependent solutions, is tailored to specific use-cases or inherently different to use. This thesis proposes the use of the Stochastic Block Model (SBM), a latent variable model for graphs, which takes difficult modeling of graphs from the user by learning the underlying structure in an unsupervised manner.

The capabilities of SBM and Degree Corrected Block Model (DCBM) to infer meaningful groups in an IP communication graph are evaluated using a graph based on ports and IP addresses (Port-Graph), and a simple IP-to-IP communication graph (IP-Graph). Both graphs are extracted from the same packet capture (PCAP) file. The groups inferred on the graphs with the different models are evaluated with respect to services consumed from hosts in different groups, topological roles of hosts in different groups, IP subnets and traffic characteristics. The results show that SBM as well as DCBM are able to extract meaningful groups. The ability of the DCBM to accommodate vertices with different degrees in one group thereby proofs as hindrance. The separation inferred with the SBM is clearer and better interpretable. Vertices are separated according to their topological role, i.e., in clients and servers. Client groups cannot be associated with one distinctive service in general. Instead, they consist of a mix of services shared by those clients. The evaluation clearly shows that groups do not correspond to specific IP-subnets.

The ability of SBM and DCBM to generate realistic synthetic graphs is successfully shown on a router level graph and the IP-to-IP communication graph from the group analysis. As before, the SBM proves superior to the DCBM with respect to inferring meaningful structure in the router level graph. The SBM infers groups to which clear topological properties can be assigned. The DCBM achieves a higher compression on the IP-to-IP graph, that is, the DCBM needs less groups to capture the structure of the graph. The groups do not reflect topological prop-

erties of the vertices, though. The DCBM is also able to generate graphs with the observed heavy tailed degree distribution. To benefit from the descriptive power and the generative capabilities of SBM and DCBM, a combined approach is proposed. This approach is able to outperform the reference generator Orbis, a state of the art network generator.

The ability to distinguish hosts in an IP-to-IP communication network based on their topological role and connectivity pattern is then leveraged to successfully detect hosts participating in a botnet in a university network. The proposed approach has a true positive rate of 100 % once the malware on infected hosts becomes active. The approach has a false positive rate of 2.5 %, which is considered rather high. As the performed analysis shows, the selected threshold to decide whether or not a host is anomalous can be increased without affecting the true positive rate, but eliminating all false alarms, i.e., resulting in a false positive rate of 0 %. The proposed method does not rely on packet inspection or malware specific signatures but instead uses easy to obtain IP connectivity information among all hosts of the network.

The possibilities for future research are manifold. Regarding synthetic graph generation, one possible direction of research is additional learning and generating of distributions over edge and vertex attributes. Another important topic is the generation of dynamic graphs, that is, graph topologies that change over time [XI13, MM15]. Regarding the separation of hosts in IP-to-IP communication networks more sophisticated versions of the SBM can be evaluated, which are capable of including metadata in their decision process [NC15]. This has the potential to improve the found separation. Also, an evaluation and implementation can be done using the separation found with the SBM to actually perform network management tasks, such as resource allocation. An important task with respect to security is an analysis of the run time of the method, i.e., if it is suitable for online detection of threats. Also, the behavior of other malware such as worms or viruses can be analyzed with the SBM. Furthermore, the applicability of the SBM to detect network wide anomalies, that is, larges scale changes in the communication structure of hosts, can be evaluated.

List of Figures

List of Tables

Glossary

DCBM A variant of the SBM allowing for heterogeneous degree distributions within communities and is thus able to model heavy tailed degree distributions. . 6, 96

DSBM Directed version of the SBM. . 46, 63, 95, 97

K-Means Algorithm to find clusters of data points in a multi-dimensional space.. 23

. 46

LBNL An IP-to-IP communication graph obtained from traces of the Lawrence Berkeley National Laboratory. 76–78, 80, 92–94

malware Short for malicious softare is a generic term for any software used to disrupt operation of communciation networks, gather sensitivie information, breach into private communication networks or display unwanted adervisement. As such, worms, viruses and botnets are all a specific instance of malware . 43, 44

Orbis Network generator preserving the degree- or joint degree distribution of an input graph . 43, 76, 82, 83, 91, 94, 110

SBM A Latent-Variable Model encoding a probability distribution over graphs . 6

Acronyms

AIC Akaike Information Criterion. 24

AS Autonomous System. 75

BIC Bayesian Information Criterion. 24

CCDF complementary cumulative distribution function. 17, 98, 100

CDF cumulative distribution function. 54, 56, 59, 63, 64, 66, 68

CI Conditional Independence. 22, 101

DAG Directed Acyclic Graph. 21, 22

DCBM Degree Corrected Block Model. 6, 8, 36, 38, 39, 45, 73, 76–82, 89, 90, 94–96, 109, 110, *Glossary:* DCBM

DDCBM Directed Degree Corrected Block Model. 38–40, 46, 51, 53, 54, 57, 59, 63, 64, 66–68, 70–74, 76–78, 80–82, 93–96, 112

DDoS Distributed Denial of Service. 77

DGM Directed Graphical Model. 21, 22, 30, 31

DSBM Directed Stochastic Block Model. 46, 63, 64, 66–68, 70–74, 76–78, 92, 94–100, 107, 112, *Glossary:* DSBM

GM Graphical Model. 20, 21

GMM Gaussian Mixture Model. 22

HOT A Router-level graph.. 76, 77, 80, 88, 89

HTTP Hypertext Transfer Protocol. 48, 57, 59, 61, 73

HTTPS Hypertext Transfer Protocol Secure. 48

iid independently and identically distribute. 80

LBNL Lawrence Berkeley National Laboratory. 46, 76, 77, *Glossary:*

LVM Latent Variable Model. 22, 23

MAD Median Absolute Deviation. 61, 68, 103, 104

MDL Minimum Description Length. 24, 25, 30, 35, 39, 40, 51, 53, 54, 63, 64, 76–78, 89, 94, 100

MLE Maximum Likelihood Estimate. 30, 36, 38, 100

NCP NetWare Core Protocol first. 59

NTP Network Time Protocol first. 48, 59, 66

P2P-Traffic Peer-to-Peer Traffic. 52

PCA Principal Component Analysis. 29, 57

PCAP Packet Capture. 45, 46, 99, 109

RV Random Variable. 21, 22, 80

SBM Stochastic Block Model. 6–8, 29–32, 34–36, 39, 76–82, 88–90, 94–96, 109, 110, 114, 117, *Glossary:* SBM

SLP Service Location Protocol. 59, 67

SNMP Simple Network Management Protocol. 48

SSH Secure Shell Protocol. 48, 61

UDP User Datagram Protocol. 59, 67

Glossary

DCBM A variant of the SBM allowing for heterogeneous degree distributions within communities and is thus able to model heavy tailed degree distributions. . 6, 96

DSBM Directed version of the SBM. . 46, 63, 95, 97

K-Means Algorithm to find clusters of data points in a multi-dimensional space.. 23

. 46

LBNL An IP-to-IP communication graph obtained from traces of the Lawrence Berkeley National Laboratory. 76–78, 80, 92–94

malware Short for malicious softare is a generic term for any software used to disrupt operation of communciation networks, gather sensitivie information, breach into private communication networks or display unwanted adervisement. As such, worms, viruses and botnets are all a specific instance of malware . 43, 44

Orbis Network generator preserving the degree- or joint degree distribution of an input graph . 43, 76, 82, 83, 91, 94, 110

SBM A Latent-Variable Model encoding a probability distribution over graphs . 6

Bibliography

[AB02] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, Jan 2002.

[ABL10] Yong-Yeol Ahn, James P. Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, August 2010.

[AGL15] Giovanni Accongiagioco, Enrico Gregori, and Luciano Lenzini. S-BITE: A Structure-Based Internet Topology gEnerator. *Computer Networks*, 77:73–89, February 2015.

[AJC15] Christopher Aicher, Abigail Z. Jacobs, and Aaron Clauset. Learning latent block structure in weighted networks. *Journal of Complex Networks*, 3(2):221–248, 2015.

[AKM+05] William Aiello, Charles Kalmanek, Patrick McDaniel, Subhabrata Sen, Oliver Spatscheck, and Jacobus Van der Merwe. Analysis of communities of interest in data networks. In *International Workshop on Passive and Active Network Measurement*, pages 83–96. Springer, 2005.

[BBPSV04] A. Barrat, M. Barthlemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Science*, 101:3747–3752, March 2004.

[ber] Enterprise tracing project.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly Detection: A Survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[CDZ97] K. L. Calvert, M. B. Doar, and E. W. Zegura. Modeling Internet topology. *IEEE Communications Magazine*, 35(6):160–163, June 1997.

[Cla] Aaaron Clauset. Network analysis and modeling, csci 5352.

[DHB+10] Guillaume Dewaele, Yosuke Himura, Pierre Borgnat, Kensuke Fukuda, Patrice Abry, Olivier Michel, Romain Fontugne, Kenjiro Cho, and Hiroshi Esaki. Unsupervised host behavior classification from connection patterns. *International Journal of Network Management*, 20(5):317–337, 2010.

[EAAT04] Daniel R. Ellis, John G. Aiken, Kira S. Attwood, and Scott D. Tenaglia. A Behavioral Approach to Worm Detection. In *Proceedings of the 2004 ACM Workshop on Rapid Malcode*, WORM '04, pages 43–53, New York, NY, USA, 2004. ACM.

[FSR09] M. Feily, A. Shahrestani, and S. Ramadass. A Survey of Botnet and Botnet Detection. In *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, pages 268–273, June 2009.

[Gar] Sebastin Garca. Ctu-malware-capture-botnet-50.

[Gru04] P. Grunwald. A tutorial introduction to the minimum description length principle. *ArXiv Mathematics e-prints*, June 2004.

[GXX15] Chen Guanrong, Wang Xiaofan, and Li Xiang. *Fundamentals of Complex Networks: Models, Structures and Dynamics*. Wiley Online Library, March 2015.

[GZC14] Sebastin Garca, Alejandro Zunino, and Marcelo Campo. Survey on Network-based Botnet Detection Methods. *Sec. and Commun. Netw.*, 7(5):878–903, May 2014.

[GZFA10] Anna Goldenberg, Alice X. Zheng, Stephen E. Fienberg, and Edoardo M. Airoldi. A survey of statistical network models. *Foundations and Trends in Machine Learning*, 2(2):129–233, 2010.

[HLL83] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[HSCZ10] Shun-Wen Hsiao, Y. S. Sun, M. C. Chen, and Hui Zhang. Behavior profiling for robust anomaly detection. In *2010 IEEE International Conference on Wireless Communications, Networking and Information Security*, pages 465–471, June 2010.

[HSM14] T. Herlau, M. N. Schmidt, and M. Mrup. Infinite-degree-corrected stochastic block model. *pre*, 90(3):032819, September 2014.

[HSS08] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008.

[HZRR15] Y. Hu, F. Zhang, K. K. Ramakrishnan, and D. Raychaudhuri. GeoTopo: A PoP-level Topology Generator for Evaluation of Future Internet Architectures. In *2015 IEEE 23rd International Conference on Network Protocols (ICNP)*, pages 90–99, November 2015.

[IGER+10] Marios Iliofotou, Brian Gallagher, Tina Eliassi-Rad, Guowu Xie, and Michalis Faloutsos. Profiling-By-Association: a resilient traffic profiling solution for the internet backbone. In *Proceedings of the 6th International COnference*, page 2. ACM, 2010.

[Ili09] Marios Iliofotou. Exploring graph-based network traffic monitoring. In *INFOCOM Workshops 2009, IEEE*, pages 1–2. IEEE, 2009.

[JGL15] A. Jakalan, J. Gong, and S. Liu. Profiling ip hosts based on traffic behavior. In *2015 IEEE International Conference on Communication Software and Networks (ICCSN)*, pages 105–111, June 2015.

[JGS+16] Ahmad Jakalan, Jian Gong, Qi Su, Xiaoyan Hu, and Abdeldime M.S. Abdelgder. Social Relationship Discovery of IP Addresses in the Managed IP Networks by Observing Traffic at Network Boundary. *Comput. Netw.*, 100(C):12–27, May 2016.

[KE02] K. Klemm and V. M. Eguluz. Growing scale-free networks with small-world behavior. *pre*, 65(5):057102, May 2002.

[Kea15] Kibae Kim et al. Effect of Homophily on Network Evolution. Technical report, Seoul National University; Technology Management, Economics, and Policy Program (TEMEP), 2015.

[KF] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models.* MIT Press.

[KN11] Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1):016107, 2011.

[LLK+13] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764 – 766, 2013.

[MABXS10] Jose Andre Morales, Areej Al-Bataineh, Shouhuai Xu, and Ravi Sandhu. Analyzing and Exploiting Network Behaviors of Malware. In Sushil Jajodia and Jianying Zhou, editors, *Security and Privacy in Communication Networks: 6th Iternational ICST Conference, SecureComm 2010, Singapore, September 7-9, 2010. Proceedings*, pages 20–34. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. DOI: 10.1007/978-3-642-16161-2_2.

[Mah] Priya Mahadevan. Analyzing and generating network topologies with orbis.

[MHK+07] Priya Mahadevan, Calvin Hubble, Dmitri Krioukov, Bradley Huffaker, and Amin Vahdat. Orbis: rescaling degree correlations to generate annotated internet topologies. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 325–336. ACM, 2007.

[MKF+06] Priya Mahadevan, Dmitri Krioukov, Marina Fomenkov, Xenofontas Dimitropoulos, k c claffy, and Amin Vahdat. The internet as-level topology: Three data sources and one definitive metric. *SIGCOMM Comput. Commun. Rev.*, 36(1):17–26, January 2006.

[MKFV06] Priya Mahadevan, Dmitri Krioukov, Kevin Fall, and Amin Vahdat. Systematic topology analysis and generation using degree correlations. In *ACM SIGCOMM Computer Communication Review*, volume 36, pages 135–146. ACM, 2006.

[MLMB01] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. BRITE: An approach to universal topology generation. In *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on*, pages 346–353. IEEE, 2001.

[MM15] C. Matias and V. Miele. Statistical clustering of temporal networks through a dynamic stochastic block model. *ArXiv e-prints*, June 2015.

[MMB00] Alberto Medina, Ibrahim Matta, and John Byers. On the origin of power laws in internet topologies. *SIGCOMM Comput. Commun. Rev.*, 30(2):18–28, April 2000.

[Mur] Kevin. P Murphy. *Machine Learning*. MIT Press.

[NC15] Mark E. J. Newman and Aaron Clauset. Structure and inference in annotated networks. *CoRR*, abs/1507.04001, 2015.

[New03] M. E. Newman. Mixing patterns in networks. *Physical Review*, 67(2):026126, February 2003.

[New10] M. E. J. Newman. *Networks: An Introduction*. Oxford University Press, March 2010.

[Pei12] Tiago P. Peixoto. Entropy of stochastic blockmodel ensembles. *Phys. Rev. E*, 85(5):056122, May 2012.

[Pei13] Tiago P. Peixoto. Parsimonious Module Inference in Large Networks. *Physical Review Letters*, 110(14), April 2013.

[Pei14] T. P. Peixoto. Hierarchical Block Structures and High-Resolution Model Selection in Large Networks. *Physical Review X*, 4(1):011047, January 2014.

[RC93] Peter J. Rousseeuw and Christophe Croux. Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, 88(424), 1993.

[RFT13] R. Rossi, S. Fahmy, and N. Talukder. A multi-level approach for evaluating internet topology generators. In *Proc. IFIP Networking Conference*, pages 1–9, May 2013.

[Roo16] Teemu Roos. *Minimum Description Length Principle*, pages 1–4. Springer US, Boston, MA, 2016.

[RTP15] M. Roughan, J. Tuke, and E. Parsonage. Estimating the Parameters of the Waxman Random Graph. *ArXiv e-prints*, June 2015.

[Run] Thomas A. Runkler. *Data Analytics*.

[SBN12] Tonmoy Saikia, Ferdous A. Barbhuiya, and Sukumar Nandi. A behaviour based framework for worm detection. *Procedia Technology*, 6:1011 – 1018, 2012.

[Sei83] Stephen B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269 – 287, 1983.

[Sta12] John Stafford. *Behavior-based Worm Detection*. PhD thesis, University of Oregon, 2012.

[TPGK03] Godfrey Tan, Massimiliano Poletto, John Guttag, and Frans Kaashoek. Role classification of hosts within enterprise networks based on connection patterns. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '03, pages 2–2, Berkeley, CA, USA, 2003. USENIX Association.

[Wax88] Bernard M. Waxman. Routing of multipoint connections. *Selected Areas in Communications, IEEE Journal on*, 6(9):1617–1622, 1988.

[WJ] Jared Winick and Sugih Jamin. Inet-3.0: Internet topology generator. Technical report.

[XI13] Kevin S. Xu and Alfred O. Hero III. Dynamic stochastic blockmodels: Statistical models for time-evolving networks. *CoRR*, abs/1304.5974, 2013.

[XWG11] K. Xu, F. Wang, and L. Gu. Network-aware behavior clustering of Internet end hosts. In *2011 Proceedings IEEE INFOCOM*, pages 2078–2086, April 2011.

[YJK+12] Xiaoran Yan, Jacob E. Jensen, Florent Krzakala, Cristopher Moore, Cosma Rohilla Shalizi, Lenka Zdeborová, Pan Zhang, and Yaojia Zhu. Model selection for degree-corrected block models. *CoRR*, abs/1207.3994, 2012.

[Zho06] Shi Zhou. Characterising and modelling the internet topology – The rich-club phenomenon and the PFP model. *BT Technology Journal*, 24(3):108–115, 2006.

[ZYM12] Yaojia Zhu, Xiaoran Yan, and Cristopher Moore. Oriented and degree-generated block models: Generating and inferring communities with inhomogeneous degree distributions. *CoRR*, abs/1205.7009, 2012.