

# Coding of Graphs with Application to Graph Anomaly Detection

Anders Høst-Madsen and June Zhang

Department of Electrical Engineering, University of Hawaii, Manoa  
Honolulu, HI, 96822, Email: {ahm,zjz}@hawaii.edu

**Abstract**—This paper has dual aims. First is to develop practical universal coding methods for unlabeled graphs. Second is to use these for graph anomaly detection. The paper develops two coding methods for unlabeled graphs: one based on the degree distribution, the second based on the triangle distribution. It is shown that these are efficient for different types of random graphs, and on real-world graphs. These coding methods are then used for detecting anomalous graphs, based on structure alone. It is shown that anomalous graphs can be detected with high probability.

## I. INTRODUCTION

A popular research problem in data mining is graph anomaly detection, which has applications in areas ranging from finance to power grid operation to detecting social trends [1], [2], [3]. In this paper we explore using description length for graph anomaly detection; that is, we encode the graph using a lossless source coder, and use the resulting codelength as the decision criteria. While minimum description length (MDL) has been used in the connection with graph anomaly detection, the application has only been for model selection in time-series analysis. As far as we know, this paper is the first to consider using description length directly for anomaly detection.

Reference [4] was the first paper to develop practical source coding algorithms for graphs. To use source coding for description length analysis, the codelength has to reflect the information in the graph, and the only information [4] reflects really is the edge probability  $p$  (see discussion later). This paper therefore develops new practical (universal) source coding algorithms based on more informative statistics. This focus is different than other recent papers in graph coding [5], [6], [7], [8] that are aimed more at entropy analysis.

### A. Graphs

The structure of a graph is defined by the set of **vertices** (also called nodes)  $\mathcal{V}$ , and the set of **edges**,  $\mathcal{E}$ . Usually, the ordering of the vertices are irrelevant, and in that case we call the graph **unlabeled**; we will only consider unweighted, unlabeled, undirected graphs in this paper. A graph,  $G(\mathcal{V}, \mathcal{E})$ , is often represented by the **adjacency matrix**,  $\mathbf{A} = [A_{ij}]$ , a  $|\mathcal{V}| \times |\mathcal{V}|$  matrix where  $A_{ij} = 1$  if  $(i, j) \in \mathcal{E}$ . The degree of a vertex is the number of edges emanating from the vertex. The **degree distribution** is the collection of the degrees of all the nodes in the graph and is an often used statistics to differentiate between different classes of random graphs such as Erdős-Rényi Barabási-Albert or Watts-Strogatz graphs [9]. There is a one-to-one correspondence between binary, symmetric

matrices and unweighted, undirected graphs, and coding of graphs is therefore equivalent to coding binary, symmetric matrices.

### B. Description Length

The description length of the data is the number of bits required to describe the data exactly: the data is turned into a stream of bits, and from this the data should be able to be recovered exactly by a decoder. We are only concerned with the length of the encoding, i.e., the number of bits output by the encoder.

The central idea here is that the description length has some relationship with the "meaning" of data. For example, Rissanen considered "useful information" in [10]. More concretely, description length can be used for data analysis. A traditional application, in particular in terms of minimum description length (MDL) [11], has been for model selection in data analysis. The methodology we will develop for graph coding can also be used for model selection for more general data sets. However, we are more interested in description length as a general data processing tool beyond simple model selection. One example is atypicality which is described in Section III.

A central principle of description length is the constraint that a decoder should be able to reconstruct the original data from an (infinite) stream of bits. One manifestation is of course the Kraft inequality [12], but the principle is more general. Since most source coding algorithms are sequential, decodability then means that the decoder can only use past decoded information to decode future data. For graphs, this is much more complicated to satisfy than for sequences. Decodability now becomes an algorithmic constraint rather than a probabilistic one, moving description length theory closer to Kolmogorov complexity [13], [12].

## II. CODING

We will base graph coding on the adjacency matrix – due to symmetry, only the lower triangular part has to be coded. However, usually the numbering of nodes is irrelevant. The resulting graph modulo automorphisms is called the structure [4]. Using this in encoding can lead to smaller codelength. Importantly, for data analysis, clearly the structure is more relevant, and description length therefore should be based on the structure.

The adjacency matrix is a binary matrix, and coding this is therefore similar to the problem considered by Steinruecken

1	1	1	...
1	1	1	...
1	1	0	...
1	0	1	...
1	0	0	...
0	1	1	...
0	1	0	...
0	0	1	...
0	0	0	...

Table I

THE FIRST COLUMN HAS ONE GROUP, THE SECOND TWO (BLUE/GREEN),  
THE THIRD FOUR (RED/CYAN/PURPLE/PINK).

in [14], on which we will base our coding. Steinruecken considered coding of unordered iid sequences, which we will think of as a matrix. We can state the approach more abstractly as follows: we first sort the rows according to some criterion (e.g., lexicographically). The coding is done on the sorted matrix, and only the sorted matrix is reproduced (exactly) at the receiver. The trick is to sort in such a way that coding of the sorted matrix is more efficient than coding the original matrix. The procedure in [14] is to first sort the sequences lexicographically (with 1 coming before 0). We say that the sequences are grouped: the first group is all sequences, the next two groups are sequences that start with 1/0, which is then subgrouped into sequences starting with 11/10/01/00, see Table I. An efficient code is as follows: we first transmit the number of ones in the first column (the first group). The next column is divided into two groups: those rows that has 1 in the first column, and those that have 0. We transmit the number of ones in each group. When the sequences are iid, the number of ones is binomially distributed, which can be used for encoding. We continue this way (with empty groups not encoded).

This approach can also be applied to adjacency matrices, with the modification that when we permute the rows during sorting, we have to do the same permutation of columns to preserve symmetry. This turns out to be equivalent to the algorithm in [4], but describing it this way reveals that the approach in [4] is strongly aimed at Erdős-Rényi graphs. From a data analysis point of view this is problematic. The only parameter the algorithm in [4] is sensitive to is the average node degree  $\bar{k}$  (equivalently  $p$ ). Consider anomalous graph detection in terms of atypicality (this is described in more detail in Section III-A): We compare the codelength of encoding the graph with a given learned coder and a universal coder. Since the only parameter [4] is sensitive to is  $p$ , this corresponds to a hypothesis test of  $p = p_0$  versus  $p \neq p_0$ . This is not irrelevant, but it is far from what we do with sequences, where we can test a given FSM against the whole class of alternative FSM. Thus, to be effective for data analysis, we need much more effective coders. In the following we will describe two such coders.

#### A. Coding Using Degree Distribution

Assume we know the degree distribution  $P(k)$ , either from a model, from learning, or from the given graph. How can we take this into account in coding? Consider coding of a given column of the sorted adjacency matrix, as outlined above. Important here is what the decoder already knows,

from previous columns: it knows the number of ones above the diagonal, it knows the number of groups  $g$ , and it knows the size  $s_i$  of each group; let  $s = \sum_{i=1}^g s_i$ . We first encode the (total) degree of the node. Call the number of ones above the diagonal  $\bar{k}$ . We can use the coding distribution

$$P(k|k \geq \bar{k}) = \frac{P(k)}{\sum_{j=\bar{k}}^{\infty} P(j)} \quad (1)$$

The decoder now has encoded the number of new ones (or edges) to encode. The encoder needs to encode which configuration of the  $k - \bar{k}$  is seen; that is, how many ones  $k_i$  are in each group, subject to the total count being  $k - \bar{k}$ . We assume that every sequence with  $k - \bar{k}$  ones is equally likely, so calculating the probability of seeing a specific configuration is just a counting problem. In total there are  $\binom{s}{k - \bar{k}}$  sequences with  $k - \bar{k}$ , and there are  $\binom{s_i}{k_i}$  ways to arrange the  $k_i$  ones in each group. The coding probability of a specific configuration therefore is

$$\log P = \sum_{i=1}^g -\log \binom{s_i}{k_i} + \log \binom{s}{k - \bar{k}}$$

A central assumption here is that at time of decoding a given column, the decoder knows the number of ones  $\bar{k}$  above the diagonal so that it can calculate (1). This is satisfied if the rows and columns are first sorted lexicographically, which can be seen as follows. Suppose  $i$  columns have been coded/decoded. The decoder knows the first  $i$  columns and rows in the (sorted) adjacency matrix: this is clearly possible to reconstruct from the number of ones in each group until column  $i$  and the fact of the sorting. The next row is chosen by the encoder among those among the remaining  $n - i$  columns that has highest sort order based on the first  $i$  columns. No matter which column is chosen, the decoder knows the first  $i$  bits, and therefore the number of ones above the diagonal.

It is not necessary to explicitly sort the adjacency matrix. Instead one can use the same partitioning algorithm from [4]. While not very explicit in the paper, they actually sort the adjacency matrix in the way they choose the next node to encode. It is seen most clearly from [4, Fig. 3].

#### B. Coding of Triangles

Edges are the most fundamental building block of graphs. A more complex building block is triangles, i.e., a cycle graph with three nodes, which is also a 3-clique. Statistics about triangles are often used to characterize graphs [9]. One statistic is the following. Consider three connected nodes  $i \leftrightarrow j \leftrightarrow k$ ; we let  $p_{\Delta}$  be the probability that there is also an edge  $i \leftrightarrow k$ . We can use this for coding as follows. Let the current node to be coded be node  $i$ , and suppose we want to code whether or not there is an edge to node  $k$ . We now look for a common neighbor  $j$  of nodes  $(i, k)$  among nodes already coded. If such a node exists, we encode the edge  $i \leftrightarrow k$  using  $p_{\Delta}$ ; otherwise, we use  $p$ . This can be used together with the structure encoding of Table I: Notice that all groups have exactly the same

connections to prior encoded nodes. Thus all the nodes  $k \in G$  in a group either has a common previously encoded neighbor with node  $i$ , or none have. Therefore, they can all be encoded with either  $p_\Delta$  or  $p$ . That is, the number of ones in the group can be encoded with a binomial distribution with probability either  $p_\Delta$  or  $p$ .

### C. Calculation and Encoding of Statistics

We consider encoding in two scenarios: learned coding, where we are given a set of training graphs and have to learn the statistics; this statistics is known both by encoder and decoder. Second, universal coding, where the encoder encodes a single graph and also has to communicate to the decoder what is the statistic.

For learned coding, the edge probability  $p$  can be estimated straightforwardly as an average. The degree distribution is estimated through a histogram. To estimate  $p_\Delta$  is more tricky. We select randomly three connected nodes  $i \leftrightarrow j \leftrightarrow k$  and calculate  $p_\Delta$  as an average. However, the value of  $p_\Delta$  depends on how the nodes are selected. When  $p_\Delta$  is used for coding, the triple of nodes is chosen in a specific way. The best estimate is therefore found by performing the coding on the training graphs. Notice that in that case the edges are divided into those coded with the triangle probability  $p_\Delta$  and those coded with  $p$ . However, those edges not (coded) in a triangle could be special. Instead of using the general  $p$ , we could estimate that  $p$  directly; we call this  $\tilde{p}_\Delta$ . In general  $p \neq \tilde{p}_\Delta$ , but in many cases they are very close.

For universal coding, there are two possible approaches, best outlined in [12, Section 13.2]: the encoder can estimate the parameters of the coding distribution and inform the decoder of the estimate. Or, the coding distribution can be sequentially calculated. For encoding  $p$  for iid coding the two approaches are essentially equivalent. The number of bits required to encode the number of ones is about  $\log \frac{n(n-1)}{2} \approx 2 \log n$  bits. For the degree distribution, we calculate the degree histogram for the whole graph, and use this for coding. The degree of a node is between 0 and  $n-1$ . We can therefore think of the degree histogram as putting each of the  $n$  (unlabeled) nodes into one of  $n$  buckets, and encoding this can be done by encoding the counts in the buckets. The number of possible configurations is a standard problem in combinatorics:  $\binom{2n-1}{n}$ , which can be transmitted with  $\log \binom{2n-1}{n} = nH\left(\frac{n}{2n-1}\right) + \frac{1}{2} \log \frac{2n-1}{n^2} + c \approx n - \frac{1}{2} \log n$  bits ( $|c| \leq 2$ ). Of course, there is a relationship between the degrees of nodes in the graph, and if we took this into consideration, it might be possible to encode the degree histogram slightly more efficient.

For triangle coding, we use sequential estimation of  $p_\Delta$  and  $\tilde{p}_\Delta$ , specifically the KT estimator [15], [16], which is  $\hat{p} = \frac{n_1 + \frac{1}{2}}{n_1 + n_0 + 1}$ , where  $n_1, n_0$  is the number of ones and zeros seen previously. The probabilities  $p_\Delta$  and  $\tilde{p}_\Delta$  are not updated after each bit, but rather after each group is encoded.

### D. Numerical Results

Some results can be seen in Fig. 1-3. In all cases, learning was done on 50 graphs prior to coding. For Erdős-Rényi graphs, the iid structure code is most efficient, but all structure codes give about the same codelength. For Barabási-Albert graphs, coding using the degree distribution is most efficient, and for Newman Watts Strogatz graphs [17], using the triangle probability is most efficient. This shows that there is no single efficient code for all graph structures.

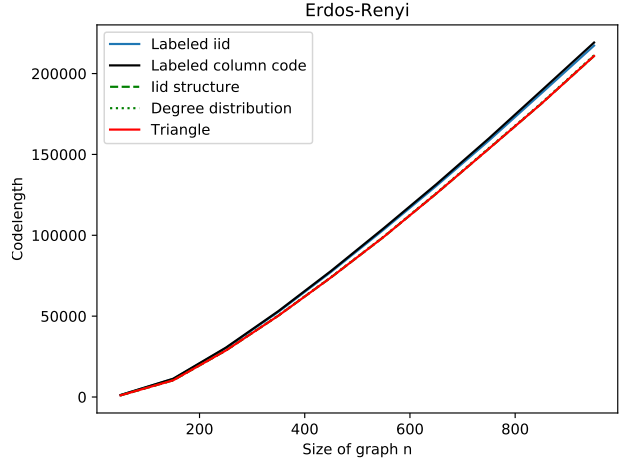


Figure 1. Comparison of different codelengths for a ER graph with  $p = \min\{\frac{1}{2}, \frac{100}{n}\}$

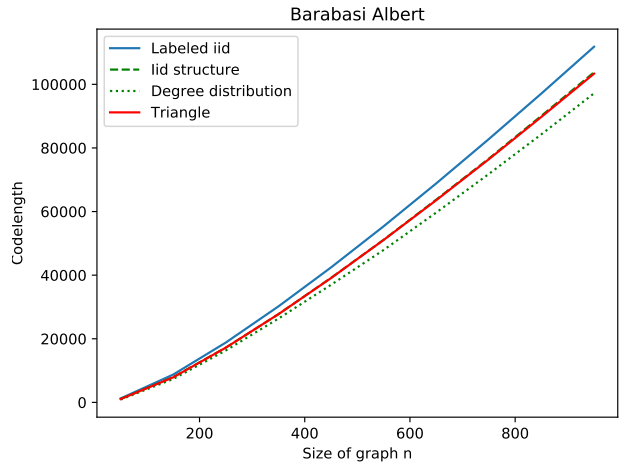


Figure 2. Comparison of different codelengths for a BA graph with  $m = 20$ .

We also did some experiments on real-world graphs, both obtained from [18]. For those graphs there is no training, so the universal coding is needed. For both graphs, using degree distribution is most efficient. However, transmitting the degree histogram is expensive, and considering that, the triangle coding is most efficient. In light of this one could consider better ways to represent the degree distribution (e.g., a parametric representation), but we have not explored that.

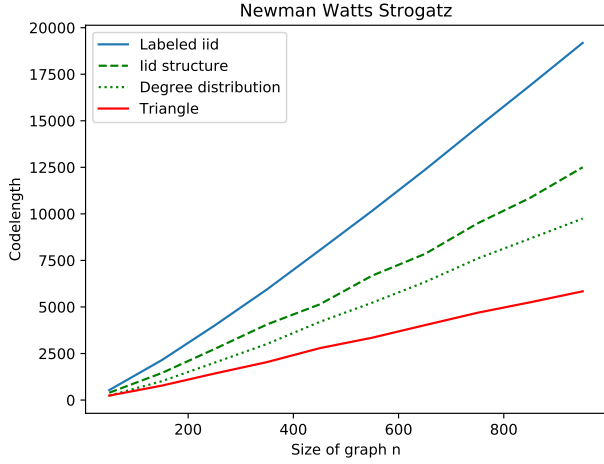


Figure 3. Comparison of different codelengths for a Watts Strogatz graph [17] with  $k=5$  and  $p=0.1$ .

Codelength	Protein graph	Power graph
Labeled iid	20513	81077
Structure iid	8796	32013
Degree distribution	7290	27651
Degree distribution with overhead	8743	32586
Triangle	8369	26507

Table II

REAL-WORLD GRAPHS. THE PROTEIN GRAPH IS THE LARGEST CONNECTED COMPONENT OF A NETWORK OF PROTEIN INTERACTIONS IN THE YEAST *SACCHAROMYCES CEREVISIAE*. THE POWER GRAPH REPRESENTS THE US WESTERN POWER GRID.

### III. ANOMALY DETECTION

For detecting anomalous graphs, we will use atypicality developed in [19], which is described by

**Definition 1.** A sequence is atypical if it can be described (coded) with fewer bits in itself rather than using the (optimum) code for typical sequences.

The papers [19], [20] show that atypicality has many desirable theoretical properties and that it works experimentally for sequences. Specifically for anomaly detection, the paper [20] shows that atypicality is (asymptotically) optimum for finite state machine (FSM). We will say that two FSM are *distinct* if they have no identical classes. Then

**Theorem 2 ([20]).** Suppose that the typical model is an FSM. Let the atypicality detector be given an anomalous sequence generated by an FSM distinct from the typical FSM. Then as the length of the sequence  $l \rightarrow \infty$ , the probability of detecting the anomaly converges to 1, while the probability of false alarm converges to 0.

As far as we know, nothing similar has been proven for any other anomaly detection methods.

#### A. Anomalous Graph Detection

In anomalous graph detection, we are given a set of training graphs  $G_1, \dots, G_T$ , and the problem is then to determine if a given graph  $G$  is anomalous based on the training. We will apply atypicality to this problem. The methodology follows directly from Definition 1. We learn coding of typical graphs,

Section II-C, and compare this with applying a universal source coder to  $G$ . In this paper, we consider unweighted, undirected graphs.

For Erdős-Rényi graphs, atypicality reduces to a hypothesis test of  $\hat{p} = p$  versus  $\hat{p} \neq p$ , which is of the form  $|\hat{p} - p| \geq \tau$  for some threshold. There is no reason to use coding, and even coding structure as in [4] does not help: in a test of  $\hat{p} = p$  versus  $\hat{p} \neq p$ , the structural decomposition would be the same, only the coding of the resulting bitstreams would be different.

For more complicated classes of random graphs such as Barabási-Albert or Watts Strogatz [17], more information can be obtained using the coding algorithms developed in Section II. The general procedure is as follows

- 1) On the set of training graphs, we run all the coding algorithms. For each we learn the values of the parameters (e.g., the histogram) for the algorithm. We choose the coder that gives the shortest codelength. The typical coder is now that algorithm with the learned parameters. Both coder and decoder know the values of the parameters, so this does not need to be encoded.
- 2) On the set of test graphs, we run first the typical coder and obtain the typical code length  $L_T$ . We then run all the coding algorithms from Section II; to each codelength we have to add the overhead of encoding the parameters (e.g., histogram). The atypical codelength,  $L_A$ , is now the minimum of these codelengths, plus a few bits to tell which coder was the shortest. The atypicality measure is the difference between the atypical codelength and the typical codelength,  $L_A - L_T$ . If  $L_A - L_T < 0$ , or is smaller than some threshold<sup>1</sup>, then following Definition 1, the graph is declared atypical (anomalous).

We tested this procedure by generating various random graphs with  $n = 100$  nodes.

The typical graphs were generated by using Barabási-Albert graphs model ( $m = 10$ ). We trained on 100 randomly generated graphs. We then generated 500 test graphs each of:

- 1) Barabási-Albert graphs ( $m = 10$ ) (i.e., typical graphs)
- 2) Barabási-Albert graphs ( $m = 9$ )
- 3) Erdős-Rényi graphs ( $p = 0.182$ ), chosen so that the graph has the same average degree as the typical graph.
- 4) Mixture graph: combination of Barabási-Albert graphs ( $m = 10$ ) and Erdős-Rényi graphs with  $p = 0.5$ ; these are essentially Barabási-Albert graphs with extra edges added ( $p$ ) to make more triangles.

We then estimated the probability density function (pdf) of the atypicality measure:  $L_A - L_T$ . The results are in Fig. 4. We can see that Erdős-Rényi and Barabási-Albert ( $m = 9$ ) test graphs can be easily distinguished from the typical graphs, Barabási-Albert ( $m = 10$ ). Identifying mixture graph from Barabási-Albert ( $m = 10$ ) is more difficult. However, due to the law of large numbers, anomaly detection improves as graph size increases. Figure 5 shows the estimated pdf of atypicality

<sup>1</sup>The threshold has a coding interpretation: it is the number of bits required to tell the decoder an atypical coder is used [19]

measures between mixture graph and Barabási-Albert ( $m = 10$ ) for graphs with  $n = 400$  nodes; if we choose the threshold to be 305, we get  $P_{\text{false alarm}} = P_{\text{miss}} = 2.4\%$ .

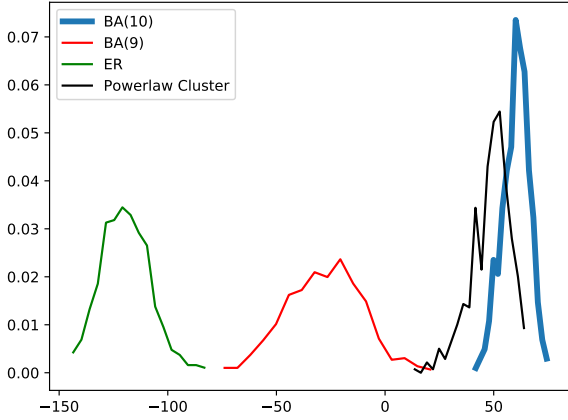


Figure 4. Pdf of atypicality measure for different types of graphs ( $n = 100$ ). The typical graphs are BA(10), which are used for training.

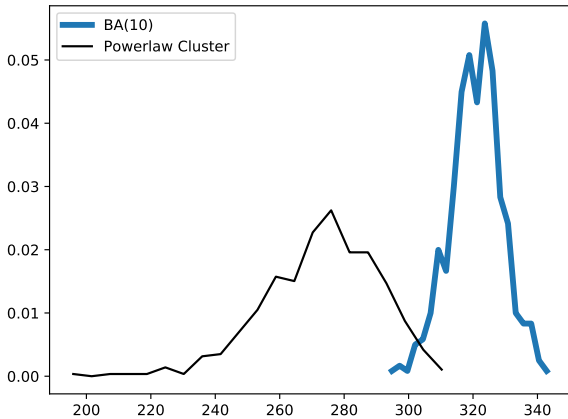


Figure 5. Pdf of atypicality measure for different types of graphs ( $n = 400$ ). The typical graphs are BA(10), which are used for training.

#### IV. CONCLUSIONS AND FUTURE WORK

In this paper we have developed a number of new universal graph coding algorithms. The minimum codelength is found by coding with each algorithm, and then finding the minimum (or weighting as in [21]). However, this is still far from the state of the art for sequences, where there are single algorithms such as CTW [16] and Lempel-Ziv [12] that can code sequences with variable complexity. One possibility is to generalize the triangle coding to consider structures of variable complexity, and weight these in an approach similar to CTW.

We have shown that the coding algorithms can be used for graph anomaly detection based on structure alone. We will

consider a number of extensions. First, in most graph-based anomaly detection problems, the anomaly is in the data on the graph. Our idea is to combine graph structure coding with coding of the data to get a single measure that takes into account both data and structure. Second, we need to be able to consider graphs of variable size; the complication here is that statistics might very well depend on size. Finally, we will consider detecting anomalous subgraphs.

#### REFERENCES

- [1] Caleb C Noble and Diane J Cook. Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2003.
- [2] William Eberle and Lawrence Holder. Anomaly detection in data represented as graphs. *Intelligent Data Analysis*, 11(6):663–689, 2007.
- [3] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2015.
- [4] Y. Choi and W. Szpankowski. Compression of graphical structures: Fundamental limits, algorithms, and experiments. *IEEE Transactions on Information Theory*, 58(2):620–638, Feb 2012.
- [5] P. Delgosha and V. Anantharam. Universal lossless compression of graphical data. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 1578–1582, June 2017.
- [6] Tomasz Luczak, Abram Magner, and Wojciech Szpankowski. Structural information and compression of scale-free graphs. on internet, 2017.
- [7] T. Luczak, A. Magner, and W. Szpankowski. Asymmetry and structural information in preferential attachment graphs. *ArXiv e-prints 1607.04102*, July 2016.
- [8] Amir R. Asadi, Emmanuel Abbe, and Sergio Verdu. Compressing data on graphs with clusters. In *2017 IEEE International Symposium on Information Theory (ISIT)*, pages 1583–1587, June 2017.
- [9] Albert-László Barabási. *Network science*. Cambridge university press, 2016.
- [10] J. Rissanen. Complexity of strings in the class of markov sources. *Information Theory, IEEE Transactions on*, 32(4):526 – 532, jul 1986.
- [11] Jorma Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, (2):416–431, 1983.
- [12] T.M. Cover and J.A. Thomas. *Information Theory, 2nd Edition*. John Wiley, 2006.
- [13] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, third edition, 2008.
- [14] C. Steinruecken. Compressing sets and multisets of sequences. *IEEE Transactions on Information Theory*, 61(3):1485–1490, March 2015.
- [15] R. Krichevsky and V. Trofimov. The performance of universal encoding. *IEEE Transactions on Information Theory*, 27(2):199–207, Mar 1981.
- [16] F. M. J. Willems, Y.M. Shtarkov, and T.J. Tjalkens. The context-tree weighting method: basic properties. *Information Theory, IEEE Transactions on*, 41(3):653–664, 1995.
- [17] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, 1998.
- [18] Timothy A. Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Trans. Math. Softw.*, 38(1):1:1–1:25, December 2011.
- [19] Anders Høst-Madsen, Elyas Sabeti, and Chad Walton. Data discovery and anomaly detection using atypicality. *IEEE Transactions on Information Theory*, submitted. Available at <http://arxiv.org/abs/1709.03189>.
- [20] Anders Høst-Madsen, Elyas Sabeti, Chad Walton, and Su Jun Lim. Universal data discovery using atypicality. In *3rd International Workshop on Pattern Mining and Application of Big Data (BigPMA 2016) at the 2016 IEEE International Conference on Big Data (Big Data 2016) Washington D.C.*, 2016.
- [21] P. A. J. Volf and F. M. J. Willems. Switching between two universal source coding algorithms. In *Data Compression Conference, 1998. DCC '98. Proceedings*, pages 491–500, 1998.