Partition and Code: learning how to compress graphs

Giorgos Bouritsas*
Imperial College London, UK
g.bouritsas@imperial.ac.uk

Andreas Loukas
EPFL, Switzerland
andreas.loukas@epfl.ch

Nikolaos Karalias
EPFL, Switzerland
nikolaos.karalias@epfl.ch

Michael M. Bronstein
Imperial College London / Twitter, UK
m.bronstein@imperial.ac.uk

Abstract

Can we use machine learning to compress graph data? The absence of ordering in graphs poses a significant challenge to conventional compression algorithms, limiting their attainable gains as well as their ability to discover relevant patterns. On the other hand, most graph compression approaches rely on domain-dependent handcrafted representations and cannot adapt to different underlying graph distributions. This work aims to establish the necessary principles a lossless graph compression method should follow to approach the entropy storage lower bound. Instead of making rigid assumptions about the graph distribution, we formulate the compressor as a probabilistic model that can be learned from data and generalise to unseen instances. Our "*Partition and Code*" framework entails three steps: first, a partitioning algorithm decomposes the graph into subgraphs, then these are mapped to the elements of a small dictionary on which we learn a probability distribution, and finally, an entropy encoder translates the representation into bits. All the components (partitioning, dictionary and distribution) are parametric and can be trained with gradient descent. We theoretically compare the compression quality of several graph encodings and prove, under mild conditions, that PnC achieves compression gains that grow either linearly or quadratically with the number of vertices. Empirically, PnC yields significant compression improvements on diverse real-world networks.

1 Introduction

Lossless data compression has been one of the most fundamental and long-standing problems in computer science. It is by now well-understood that the intrinsic limits of compression are governed by the entropy of the underlying data distribution [1]. Crucially, these limits expose an intimate connection between compressibility and machine learning: the better one models the underlying data distribution (from limited observations) the more bits can be saved and vice-versa [2].

The compression of ordered data such as text, images, or video, underpins the modern technology from web protocols to video streaming. However, graph-structured data remain a notable exception. As graph data are becoming more prevalent, it becomes increasingly important to invent practical ways to encode them parsimoniously.

There are three main challenges one faces when attempting to compress graphs:

C1. Dealing with graph isomorphism (GI). A key difficulty that distinguishes graphs from conventional data lies in the absence of an inherent ordering of the graph vertices. In order to be able to

*This work was done while Giorgos Bouritsas was visiting Dr. Andreas Loukas at EPFL, Lausanne.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

approach the storage lower bounds, isomorphic graphs should be encoded with the same codeword. However, since the complexity of known algorithms for GI is super-polynomial on the number of vertices [3], a direct use of GI is impractical for graphs consisting of more than a few hundred vertices. Indeed, an examination of the graph compression literature reveals that most progress has been made by optimising a vertex ordering and adapting methods originally invented for vectored data [4–6]. Unfortunately, naively encoding graphs as vectors results in a significant loss in compression[2].

C2. Evaluating the likelihood. An optimal encoder [1] requires one to accurately estimate and evaluate the probabilities of all the possible outcomes of the underlying domain. When dealing with high-dimensional data, this can be addressed by partitioning the data into parts to obtain a decomposition of the probability distribution: e.g., images can be compressed by modelling the distribution of pixels or patches [10–12], and text by focusing on characters or n-grams [13–19]. However, since graphs do not admit an efficiently computable canonical ordering, it is unclear what decomposition one should employ.

C3. Accounting for the description length of the learned model. The classical learning theory trade-off between model complexity and generalisation is of paramount concern for effective compression. Though in typical deep learning applications one can aim to model the data distribution with an overparametrised neural network (NN) that generalises well, utilising such models to compress information is problematic: since decoding is impossible unless the decoder also receives the learned model (i.e., the NN parameters), *overparametrised models are, by definition, suboptimal*. This is a pertinent issue for likelihood-based neural approaches as overparametrisation is commonly argued to be a key component of why NNs can be trained [20–22].

The Partition and Code (PnC) framework. Our main contribution is PnC, a framework for learning compression algorithms suitable for encoding graphs sampled from an underlying distribution. In the heart of PnC lie two ideas: *(a) Learn to break the problem into parts.* Rather than predicting directly the likelihood, we aim to learn how to decompose graphs into non-overlapping subgraphs, see Fig. 1. *(b) Identify and code recurring subgraphs when possible.* We use a learned *dictionary* to code subgraphs that appear frequently, whereas rare subgraphs are encoded separately. The dictionary is restricted to contain only a finite number of small recurring subgraphs. This biases the model towards interpretable and well generalising solutions. Both the "partition" and "code" components of PnC are learned directly from the data, by optimising the total description length.

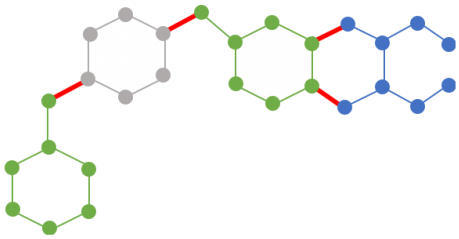

Figure 1: Illustration of the graph decomposition. The subgraph colours correspond to dictionary atoms a_1, a_2 and a_3. Cuts are denoted in red.

Our framework provides a solution to the three challenges of graph compression. C1: By constraining the dictionary to only contain graphs of size up to a small constant, we can efficiently solve GI. C2: Graph partitioning provides us with the desirable decomposition of the distribution. Using appropriate parametrised probabilistic models, we obtain a closed-form expression for the likelihood that can be later used by any close-to-optimal entropy coder [23–25]. C3: Since we use NNs to *decompose* the distribution (and not to predict the likelihood), we can rely on overparametrisation without having to relay the NN parameters to the decoder. Also, the complexity of our learned hypothesis (the dictionary) can be computed, and thus optimised, during training.

Theoretical results. Our analysis reveals that PnC can significantly improve upon less sophisticated graph encoders and justifies the usefulness of both the "Partition" and the "Code" component. Specifically, we prove that under mild conditions on the underlying graph distribution, PnC requires in expectation $\Theta(n^2)$ less bits than standard graph encodings, even if the latter are given access to an oracle that solves GI. Further, the dictionary induces additional savings of $\Theta(n)$ bits, with the gain

[2]This follows by a simple counting argument: there are $2^{\binom{n}{2}}$ labelled undirected graphs while the respective number for unlabelled graphs is asymptotically equal to $2^{\binom{n}{2}}/n!$ [7]. Thus, if all graphs of n vertices are equally probable, an encoding that does not consider isomorphism would sacrifice $\log n!$ bits [8, 9].

being inversely proportional to the entropy of the distribution of the dictionary atoms. Thus, the more repetitive the patterns in the graph distribution are, the larger will be the compression benefits of PnC.

Practical algorithms. We instantiate of our framework using the following algorithmic modules: (a) a low-parameter learnable estimate of the probability distribution, (b) learning to select the dictionary from a graph universe of finite size, and (c) learning to partition. The latter is a parametric randomised iterative algorithm, the probabilities of which are inferred from a Graph Neural Network (GNN) and optimised with reinforcement learning. Importantly, all algorithms can be jointly trained in order to minimise the total description length in a synergistic manner. We evaluate our framework on diverse real-world graph distributions and showcase compression gains with respect to both conventional and advanced baseline compressors, in observed and unseen data.

2 Related work

Engineered codecs. The majority of graph compressors are not probabilistic, but rely on hand-engineered encodings optimised to take advantage of domain-specific properties of e.g., Web-Graphs [4], social networks [6, 26, 27], and biological networks [28]. A common idea in these approaches is *vertex reordering* [4–6, 26, 27, 29–31], where the adjacency matrix is permuted in such a way that makes it "compression-friendly" for mainstream compressors of sequences, such as gzip. The algorithms identifying the re-orderings are usually based on heuristics taking advantage of specific network properties, e.g., community structure. Another recurrent idea is to detect or use predefined *frequent substructures* (e.g., cliques) to represent more efficiently different parts of the graph via grammar rules [32]. These approaches do not attempt to model the underlying graph distribution and thus to approach the storage lower bounds, but strive to find a balance between compression ratios and fast operations on the compressed graphs. Thus, despite their practical importance, they are less relevant to our work. A comprehensive survey can be found in [33].

Theory-driven approaches. Several works have contributed to the foundations of the information content and the complexity of graphs [8, 9, 34–40]. However, few works have attempted to model the underlying graph distribution. Perhaps the most outstanding progress has been made for graphs modelled by the Stochastic Block Model (SBM) [41–48]. Although originally invented for clustering and network analysis purposes, these approaches can be seamlessly used for compression due to their exact likelihood computation. In fact, as we argue in this work, virtually any graph clustering algorithm can be used successfully for compression, by defining codewords corresponding to a community-based random graph model. However, as our experiments confirm, such approaches are less effective at compressing graphs that do not contain clusters.

Likelihood-based neural approaches. Any generative model that can provide likelihood estimates in a finite sample space can be used for lossless compression. As a result, a plethora of likelihood-based neural compressors have been recently invented, ranging from autoregressive models for text [18, 19, 49, 50] and images [11, 12] to latent variable models [51–55] (paired with bits-back coding [56, 57] and Asymmetric Numeral Systems - ANS [25]), normalising flows [58–61] and most recently, diffusion-based generative models [62, 63]. However, the vast majority of current graph generators lack the necessary theoretical properties an effective graph compressor should have: they compute probabilities on labelled graphs instead of isomorphism classes by resorting to a heuristic ordering [64–71] (in general this will be suboptimal unless we canonicalise the graph/solve graph isomorphism, while different orderings will have non-zero probabilities, hence we will incur compression losses), and/or do not provide a likelihood [72–76].

An important caveat, which is often ignored in the literature, is that when parametrising the distribution with a neural network, the data cannot be recovered unless the decoder has access to the network itself. Hence it is necessary to account for the NN's description length when evaluating compression gains. However, generative models are usually parameter inefficient, while compressing them (especially during training) is a challenging problem [77]. Note that this is significantly different than compressing neural classifiers, since the capacity to infer the likelihood up to high precision needs to be retained. Therefore, these approaches can have diminishing (or even negative) returns when the dataset size is not large enough. In contrast, by optimising the total description length (equivalent to *maximum a-posteriori*), we design a compressor that is practical even for small datasets, while the learned dictionary makes our compressor interpretable.

Other related work includes using compression objectives paired with heuristic algorithms for downstream tasks, such as motif finding [78, 79] and graph summarisation [80, 81], lossy compression/coarsening [82–87], and graph dictionary learning in the context of sparse coding [88, 89].

3 Preliminaries

We use capital letters for sets A, B, bold font for vectors $\mathbf{x}$ and matrices $\mathbf{X}$, and calligraphic font $\mathcal{X}$ for families of sets. We express the information content in bits by using a base-2 logarithm $\log$.

Graphs. Let $G = \{V, E\}$ be a graph with n vertices $V = \{v_1, \ldots, v_n\}$ and m edges, where $e_{ij} \in E$ whenever vertices $v_i, v_j \in V$ are connected by an edge. For simplicity, we assume the graphs to be undirected, though the same methods apply to directed graphs as well with minimal modifications. Let $\mathrm{cut}(A, B) = \{e_{ij} \in E \mid v_i \in A \text{ and } v_j \in B\}$ denote the set of edges with endpoints at two different vertex sets $A, B \subseteq V$.

Information theory. Following the usual terminology in information theory and Minimum Description Length (MDL) theory [2, 90, 91], we assume an *observation space* $\mathfrak{G}$ (in our case a space of possible graphs), and a probability distribution p (sometimes referred to as *probabilistic source*) producing samples from the observation space. We observe a dataset $\mathcal{G} = \{G_1, G_2, \ldots, G_{|\mathcal{G}|}\}$ of i.i.d. observations drawn from p. Note that this setting is in contrast with most works on graph compression [4, 30, 5, 6], where the target is to compress a *single* large network, such as a social network or a web graph. Let a *description method* or *symbol code* $\mathsf{CODE} : \mathfrak{G} \to \{0, 1\}^*$ be a mapping from the observation space to a variable-length sequence of binary symbols, the output of which is a *codeword*.

In the context of graph compression, we are interested in the description length of the code $\mathrm{L}_{\mathsf{CODE}}(G)$, or $\mathrm{L}(G)$ for brevity, i.e., the number of bits needed to encode the graph G, rather than the code itself, given a single requirement: the code needs to be *uniquely decodable*, meaning that any concatenation of codes can be uniquely mapped to a sequence of observations. This property can be easily verified using only the description lengths by the well-known *Kraft–McMillan inequality* (l.h.s. in formula (1)), an important implication of which is that every uniquely decodable code implies a probability distribution $q(G)$ (r.h.s):

$$\sum_{G \in \mathfrak{G}} 2^{-\mathrm{L}(G)} \leq 1 \ \text{(code)} \iff q(G) = \frac{2^{-\mathrm{L}(G)}}{\sum_{G \in \mathfrak{G}} 2^{-\mathrm{L}(G)}} \ \text{(distribution)} \tag{1}$$

This equivalence allows us to always define underlying probabilistic models when working directly with description lengths. When the Kraft–McMillan inequality holds with equality, we say that the code is *complete*; the case of strict inequality means the code is *redundant*. Moreover, in the case of complete codes, the expected length of the code corresponds to the distribution entropy $\mathbb{H}_q[G]$ (plus a constant term when the code is redundant). Thus, compression is synonymous to defining probabilistic models with the lowest possible entropy. An important notion that frequently appears in our theoretical analysis is the *binary* entropy, i.e., the entropy of a bernoulli variable with probability of success p. To distinguish it from the general notion of entropy we will denote it as $\mathrm{H}(p) = -p \log p - (1-p) \log(1-p)$. It holds that $0 \leq \mathrm{H}(p) \leq 1$, where the l.h.s equality is satisfied for $p \in \{0, 1\}$ and the r.h.s. for $p = 1/2$.

Common graph encodings. An important principle that we follow is that whenever we cannot make any assumptions about an underlying distribution, or modelling it is impractical, then the uniform distribution p_{unif} is chosen for encoding. The reason is that uniform distribution is *worst-case optimal* [90]: for any non-uniform unknown distribution p, there always exists a distribution q the corresponding encoding of which will be on expectation worse than the uniform encoding: $\mathbb{E}_{x \sim p}[-\log q(x)] > \mathbb{E}_{x \sim p}[-\log p_{\text{unif}}(x)]$. In the context of graphs, when we cannot make any assumptions or when enumeration is impossible, we will be using a slightly more informative distribution: the Erdős–Rényi (ER) random graph model. This model assigns equal probability to all labelled graphs with n vertices and m edges. Assuming a uniform probability over the possible number of vertices n and number of edges m given n, we get:

$$L_{\text{null}}(G) = \log(n_{\max} + 1) + \log\left(\binom{n}{2} + 1\right) + \log\binom{\binom{n}{2}}{m}, \tag{2}$$

where $n_{\max}$ is an upper bound on the number of vertices. The "*null*" encoding compresses more efficiently graphs that are either very sparse or very dense (low-surprise) as the number of possible graphs with m edges is maximised when $m = n(n-1)/4$.

4 The Partition & Code (PnC) graph compression framework

Our pipeline consists of three main modules: a *partitioning* module, a *dictionary* module, and an *entropy encoding* module. (a) The partitioning module is responsible for decomposing the graph into disjoint subgraphs and cross-subgraph edges (or *cuts*). Subgraphs play the role of elementary structures, akin to characters or words in text compression, and pixels or patches in image compression. (b) The dictionary is a small collection of subgraphs (*atoms*) that are recurrent in the graph distribution. The dictionary module maps the partitioned subgraphs to atoms in the dictionary, allowing us to represent the graph as a collection of atom indices and cuts. (c) Finally, this representation is given as input to the an entropy encoder that translates it into bits.

Decoding the graph in a lossless way involves inverting these three steps: initially the atom indices and cuts are decoded using the same probabilistic model with the encoder, then the atoms are retrieved from the dictionary, and finally all elements are composed back to obtain the original structure. The composition becomes possible by making sure that the cuts are encoded w.r.t. an arbitrarily chosen ordering of each atom's vertices, hence the decoded graph is guaranteed to be isomorphic to the input, but not necessarily with the same vertex ordering.

4.1 Step 1. Partitioning

The first step of PnC is to employ a parametric partitioning algorithm PART_θ to decompose each graph G into b subgraphs of bounded size:

$$\mathsf{PART}_\theta(G) = (\mathcal{H}, C) \quad \text{and} \quad \mathcal{H} = \{H_1, H_2, \cdots, H_b\}, \tag{3}$$

where $H_i = \{V_i, E_i\}$ is the i-th subgraph and $C = \{V, E_C\}$ is a b-partite graph containing all cut edges $E_C = E - \cup_i E_i$. Variable θ indicates the learnable parameters.

Is partitioning necessary? In order to convert probability estimates to codewords, entropy encoders [23–25] need to be able to compute the probability of *every graph in the observation space* (or equivalently to have access to the cumulative distribution function (c.d.f.)) - see challenge C2 in Section 1. To achieve this goal, one needs to partition the observation space in a way that permits efficient enumeration of the possible outcomes. Analogously, as we will see in Section 4.3, graph partitioning allows us to decompose the distribution and thus to obtain a closed-form expression for the likelihood. Further, Theorem 1 suggests that partitioning brings a useful inductive bias for graph data providing significant storage gains compared to distribution-agnostic baselines.

4.2 Step 2. Graph dictionary

Rather than naively compressing each subgraph in (3) under a null model, an effective compression algorithm should exploit regularities in the output of PART_θ. We propose to utilise a dictionary that stores the most commonly occurring subgraphs. Concretely, we define a dictionary D to be a collection of connected subgraphs (or *atoms*) from some universe $\mathfrak{U}$:

$$D = \{a_1, a_2, \cdots, a_{|D|}\}, \quad \text{where} \quad a_i \in \mathfrak{U}. \tag{4}$$

There are two viable choices for the atom encoding: (a) If the universe is small enough to be efficiently enumerable then we can assume a uniform distribution over $\mathfrak{U}$ which yields the description length $\mathrm{L}(D) = |D| \log |\mathfrak{U}|$. Intuitively, this would amount to storing the index of each atom within a list enumerating $\mathfrak{U}$. (b) On the other hand, when $\mathfrak{U}$ is too large to enumerate the atoms can be stored one-by-one based on the null-model encoding given in (2):

$$\mathrm{L}(D) = \sum_{a_i \in D} \mathrm{L}_{\text{null}}(a_i). \tag{5}$$

It is important to note that, to be as effective as possible, the partitioning and the dictionary should be co-designed: D should capture those subgraphs that are most likely, whereas PART_θ should be biased towards subgraphs with similar structure. The use of a dictionary makes it possible to explicitly account for (and thus optimise) the description length of the learned hypothesis (i.e., equation above), which is an essential component of any compression algorithm (see challenge C3 in Section 1).

4.3 Step 3. Graph encoding

The last step entails compressing G by encoding the output $(\mathcal{H}, C)$ of PART_θ. As discussed in Section 3, a uniquely decodable code implies a probability distribution, i.e., $q_\phi(G) = q_\phi(\mathcal{H}, C)$ which corresponds to a description length $\mathrm{L}_\phi(\mathcal{H}, C) = -\log q_\phi(\mathcal{H}, C)$, where ϕ denotes the learnable parameters. In the following we explain how q_ϕ is decomposed.

Subgraphs. We opt for a dual encoding of subgraphs: one for the subgraphs that belong to the dictionary $\mathcal{H}_{\text{dict}}$, and one for non-dictionary subgraphs $\mathcal{H}_{\text{null}}$ that are encoded with the help of a null model as in Eq. (2). This choice has a dual purpose: (a) It allows PART_θ to choose non-dictionary atoms. This is crucial to our approach, since constraining the partitioning to specific isomorphism classes would significantly complicate optimisation. (b) Further, it enables us to maintain a balance between two common structures found in real-world networks—*frequent subgraphs* (stored in the dictionary), and *low-entropy subgraphs* as implied by the null model (i.e., very sparse or very dense subgraphs). The distribution is therefore decomposed into the following components:

Number of subgraphs. First, we encode the number of dictionary and non-dictionary subgraphs (b_{dict} and b_{null} respectively) as follows:

$$q_\phi(b_{\text{dict}}, b_{\text{null}}) = \text{Binomial}(b_{\text{dict}}|b; \phi) q_\phi(b) = \binom{b}{b_{\text{dict}}} (1-\delta_\phi)^{b_{\text{dict}}} \delta_\phi^{b-b_{\text{dict}}} q_\phi(b), \tag{6}$$

where $1 - \delta_\phi = \mathbb{P}[H \in D]$ is the probability of an arbitrary subgraph to belong in the dictionary and $q_\phi(b)$ is a categorical.

Dictionary subgraphs. The dictionary subgraphs are encoded in a permutation invariant way via a *multinomial* distribution, i.e., we encode the histogram of atoms:

$$q_\phi(\mathcal{H}_{\text{dict}}|b_{\text{dict}}, D) = \text{Multinomial}(b_1, b_2, \ldots, b_{|D|} \mid b_{\text{dict}}; \phi) = b_{\text{dict}}! \prod_{a \in D} \frac{q_\phi(a)^{b_a}}{b_a!}, \tag{7}$$

where $b_a = \left|\{H \in \mathcal{H}_{\text{dict}} | H \cong a\}\right|$ and $\sum_{a \in D} b_a = b_{\text{dict}}$.

Non-Dictionary subgraphs. The non-dictionary subgraphs are encoded independently according to the null model:

$$q_\phi(\mathcal{H}_{\text{null}}|b_{\text{null}}, D) = \prod_{H_i \in \mathcal{H}_{\text{null}}} q_{\text{null}}(H_i). \tag{8}$$

Cuts. We encode the cuts conditioned on the subgraphs, using a non-parametric uninformative null model for multi-partite graphs similar to [45] (see Appendix B.1 for the detailed expression) that prioritises low-entropy cuts. In this way we give more emphasis to the subgraphs and an inductive bias towards distinct clusters in the graph.

Overall, the description length of $(\mathcal{H}, C) = \mathsf{PART}_\theta(G)$ is given by

$$\mathrm{L}_\phi(\mathcal{H}, C|D) = \mathrm{L}_\phi(b_{\text{dict}}, b_{\text{null}}) + \mathrm{L}_\phi(\mathcal{H}_{\text{dict}}|b_{\text{dict}}, D) + \mathrm{L}_{\text{null}}(\mathcal{H}_{\text{null}}|b_{\text{null}}, D) + \mathrm{L}_{\text{null}}(C|\mathcal{H}), \tag{9}$$

and the learnable parameter set is $\left\{\delta_\phi, \{q_\phi(b)\}_{b=b_{\min}}^{b_{\max}}, \{q_\phi(a)\}_{a \in D}\right\}$.

Remarks about graph isomorphism (GI). Observe that given a fixed decomposition, our parametrisation is invariant to isomorphism, which is a desirable property since isomorphic graphs will be assigned codewords with the same length. Note that this is not sufficient to guarantee that all isomorphic graphs will be assigned *the same* codeword. Since this would imply a solution to GI, it remains an open problem. However, our graph encoding provides desirable tradeoffs between the expressivity of the probabilistic model, its number of parameters, and computational complexity, since it can adapt to different graph distributions using only a few parameters and solving GI only for small graphs.

4.4 Selecting a hypothesis by minimising the total description length

Putting everything together, in order to encode a graph dataset $\mathcal{G}$ sampled i.i.d. from $\mathfrak{G}$ we minimise the total description length

$$\min_{\theta, D, \phi} \sum_{G \in \mathcal{G}} \mathrm{L}_\phi(\mathsf{PART}_\theta(G) \mid D) + \mathrm{L}(D) \tag{10}$$

with respect to the parameters θ of the parametric partitioning algorithm, the dictionary D, and the parameters ϕ of the probabilistic model. Eq. (10) is a typical two-part Minimum Description Length (MDL) objective [92, 90]. Using standard MDL terminology, the tuple (θ, D, ϕ) is a *point hypotheses* and minimising (10) amounts to finding the simplest hypothesis that best describes the data.

5 Theoretical analysis: quadratic and linear gains

The following section performs a comparative analysis of the description length growth rate of various graph compressors. We compare PnC against two strong baselines: (a) The code length L_{part} induced by a pure partitioning-based graph encoding. Here, a graph is decomposed into subgraphs and cuts, but the distribution of subgraphs is not modelled (i.e., both subgraphs and cuts are encoded with a null model $\mathrm{L}_{\text{part}}(G) = \mathrm{L}_{\text{null}}(\mathcal{H}) + \mathrm{L}_{\text{null}}(C|\mathcal{H})$). (b) The code length of encodings that do not rely on partitioning but encode each graph as a whole. Importantly, our results hold even for those baselines that encode the isomorphism class of each graph, rather than the graph itself, such as the Erdős-Renyi model for unlabelled graphs of n vertices: $\mathrm{L}_{\text{ER-S}}(G) = \log|\mathfrak{G}_{n,m}| + \log(n^2+1)$, where $\mathfrak{G}_{n,m}$ is the set of all graphs with n vertices and m edges. $\mathrm{L}_{\text{ER-S}}(G)$ serves as lower bound to typical encodings such as that of Eq. (2), but can be impractical to implement due to the complexity of GI. The analysis of additional baselines can be found in Appendix A.

Our main theorem shows that, under mild conditions on the underlying graph distribution, the expected description lengths of the compared encodings are totally ordered:

Theorem 1. *Consider a distribution p over graphs with n vertices and a partitioning algorithm that decomposes a graph into b blocks of $k = O(1)$ vertices. Then it holds that:*

$$\mathbb{E}_{G\sim p}[\mathrm{L}_{\text{PnC}}(G)] \overset{(1b)}{\lesssim} \mathbb{E}_{G\sim p}[\mathrm{L}_{\text{part}}(G)] \overset{(1a)}{\lesssim} \mathbb{E}_{G\sim p}[\mathrm{L}_{\text{ER-S}}(G)] \tag{11}$$

under the following conditions:

(1a) $\frac{\log(k^2+1)}{k^2}) + \bar{\mathrm{H}}_{m_{ij}} < \bar{\mathrm{H}}_m$*, where* $\bar{\mathrm{H}}_{m_{ij}} = \mathbb{E}_{G\sim p}[\mathrm{H}(\frac{m_{ij}}{k^2})]$ *and* $\bar{\mathrm{H}}_m = \mathbb{E}_{G\sim p}[\mathrm{H}(\frac{m}{n^2})]$ *is the expected binary entropy of the cut size m_{ij} between two subgraphs and that of the total number of edges m, respectively.*

(1b) $|D| < (k^2+1)2^{k^2\bar{\mathrm{H}}_{m_i}}$*, where $|D|$ is the size of the dictionary and* $\bar{\mathrm{H}}_{m_i} = \mathbb{E}_{G\sim p}[\mathrm{H}(\frac{m_i}{k^2})]$ *the expected binary entropy of the number of edges m_i in a subgraph.*

The compression gains are:

$$\mathbb{E}_{G\sim p}[\mathrm{L}_{\text{Part}}(G)] \lesssim \mathbb{E}_{G\sim p}[\mathrm{L}_{\text{ER-S}}(G)] - n^2\Big(\bar{\mathrm{H}}_m - \frac{\log(k^2+1)}{k^2} - \bar{\mathrm{H}}_{m_{ij}}\Big) \tag{12}$$

and

$$\mathbb{E}_{G\sim p}[\mathrm{L}_{\text{PnC}}(G)] \lesssim \mathbb{E}_{G\sim p}[\mathrm{L}_{\text{part}}(G)] - nk(1-\delta)\left(\bar{\mathrm{H}}_{m_i} - \frac{\mathbb{H}(D) - \log(k^2+1)}{k^2}\right), \tag{13}$$

where $1-\delta$ is the probablity that a subgraph belongs in the dictionary and $\mathbb{H}(D) = \mathbb{H}_{a\sim q_\phi(a)}[a]$ is the entropy of the distribution on dictionary atoms $q_\phi(a)$.

Theorem 1 provides insights on the compressibility of certain graph distributions given their structural characteristics. In particular, we can make the following remarks: (a) Condition (1a) can be satisfied even for very small values of k as long as the graphs possess community structure. Perhaps counter-intuitively, when $k = O(1)$ we can satisfy the condition even if the communities have $O(n)$ size by splitting them into smaller subgraphs. This is possible because, in contrast to the majority of graph partitioning objectives that are based on minimum cuts, the compression objective attains its minimum when the cuts have "low entropy". Since communities that are tightly internally connected have large cuts, $\bar{\mathrm{H}}_{m_{ij}}$ and the code length will be kept small. This is a key observation that strongly motivates the use of partitioning for graph compression. (b) Condition (1b) provides an upper bound to the size of the dictionary, which can be easily satisfied for moderately small values of k. More importantly, the dependence of the compression gain on the entropy $\mathbb{H}(D)$, reveals that dictionary atoms should be frequent subgraphs in the distribution, confirming our intuition. The bounds also show that, since the probabilities of the atoms are estimated from the data, PnC does not need to make assumptions about the inner structure of the subgraphs and can adapt to general distributions.

In Appendix A.4, we also provide theoretical evidence on the importance of of encoding dictionary subgraphs as isomorphism classes instead of adjacency matrices, which is related to challenge C1 mentioned in the introduction. In particular, Theorem 2 shows that, if isomorphism is not taken into account, the number of bits that will be lost will grow linearly with the number of vertices. All proofs and detailed assumptions can be found in Appendix A.

6 Optimisation and learning algorithms

We turn our focus to learning algorithms for the optimisation of the MDL objective (10). The following sections explain how each parametric component of PnC is learned.

Subgraph encoding ϕ. The graph encoding is parametrised as follows: $q_\phi(a_i)$ and $q_\phi(b)$ are parametrised by learnable variables that are converted into categorical distributions over the dictionary atoms and the number of vertices respectively, using a softmax function. Similarly, δ_ϕ is parametrised by a learnable variable converted to a probability via the sigmoid function.

Dictionary D. Let $\mathfrak{U} = \{a_1, a_2, \dots a_{|\mathfrak{U}|}\}$ be a practically enumerable universe and define $\mathbf{x} = (x_1, x_2, \dots, x_{|\mathfrak{U}|})$ as

$$x_i = \begin{cases} 1 & \text{if } a_i \in D \\ 0 & \text{otherwise.} \end{cases}$$

Thus, x_i indicates whether D contains subgraph a_i. Now, optimising w.r.t the dictionary amounts to finding the binary assignments for $\mathbf{x}$ that minimise (10). To circumvent the combinatorial nature of this problem, we apply the continuous relaxation $\hat{x}_i = \sigma(\psi_i), \forall i \in \{0, 1, \dots, |\mathfrak{U}|\}$, where σ is the sigmoid, ψ_i are learned continuous variables, and $\hat{x}_i \in [0, 1]$ a fractional alternative to x_i. Appendix B.3 shows how (10) can be re-written w.r.t. $\mathbf{x}$ and optimised by using the surrogate gradient w.r.t $\hat{x}_i$.

It is important to note that, in practice, we do not have to introduce indicator variables for the entire universe: Since most subgraphs a_i will be never encountered in the graph distribution, we build the universe adaptively during training, by progressively adding the different graphs that the partitioning algorithm yields. We also allow the universe to contain subgraphs of size up to $k = O(1)$, in order to ensure that the isomorphism testing between atoms and subgraphs can be efficiently computed.

Parametric graph partitioning algorithm. Finding the graph partitions that minimise (10) in principle requires searching in the space of partitioning algorithms. Instead, we constrain this space via a differentiable parametrisation that allows us to perform gradient-based optimisation. Currently, learning to partition is an open problem, as to the extent of our knowledge known neural approaches require a fixed number of clusters [93–95] or do not guarantee that the subgraphs are connected [96].

Our *Neural Partitioning* is a randomised algorithm parametrised with a graph neural network (GNN). When run on a graph, the GNN outputs a random $(\mathcal{H}, C)$ together with a corresponding probability $p_\theta^{\text{GNN}}(\mathcal{H}, C|G)$ and training is performed by estimating the gradients w.t.t. θ with REINFORCE [97]. Our algorithm proceeds by iteratively sampling (and removing) subgraphs from the graph until it becomes empty. At each step t we select a subgraph H_t, by first sampling its vertex count k_t, and subsequently sampling at most k_t vertices. To guarantee connectivity, we also sample the vertices iteratively and mask-out the probabilities outside the pre-selected vertices' neighbourhoods. The complexity of the algorithm is $O(n)$, where n the number of the vertices of the graph. Please refer to Appendix B.4 for an in-depth explanation of the algorithm and relevant implementation details. We stress that we mainly consider this algorithm as a proof of concept that we ablate against other non-parametric partitioning algorithms. A plethora of solutions can be explored in a parametric setting and we welcome future work in this direction.

MDL objective. Given dataset $\mathcal{G}$, we train all components by minimising the description length:

$$L(\mathcal{G}) = \mathrm{L}_{\mathbf{x}}(D) + \sum_{G \in \mathcal{G}} \mathbb{E}_{(\mathcal{H}, C) \sim p_\theta^{\text{GNN}}(\mathcal{H}, C|G)} [\mathrm{L}_{\phi, \mathbf{x}}(\mathcal{H}, C|D)]. \tag{14}$$

Taking the expectation over the GNN output $(\mathcal{H}, C) \sim p_\theta^{\text{GNN}}(\mathcal{H}, C|G))$, we calculate the gradients as: $\nabla_\phi L(\mathcal{G}) = \sum_{G \in \mathcal{G}} \mathbb{E}[\nabla_\phi \mathrm{L}_{\phi, \mathbf{x}}(\mathcal{H}, C)|D)]$, $\nabla_{\hat{\mathbf{x}}} L(\mathcal{G}) = \nabla_{\hat{\mathbf{x}}} \mathrm{L}_{\hat{\mathbf{x}}}(D) + \sum_{G \in \mathcal{G}} \mathbb{E}[\nabla_{\hat{\mathbf{x}}} \mathrm{L}_{\phi, \hat{\mathbf{x}}}(\mathcal{H}, C|D)]$, and $\nabla_\theta L(\mathcal{G}) = \sum_{G \in \mathcal{G}} \mathbb{E}[\mathrm{L}_{\phi, \mathbf{x}}(\mathcal{H}, C|D) \nabla_\theta \ln p_\theta^{\text{GNN}}(\mathcal{H}, C|G)]$.

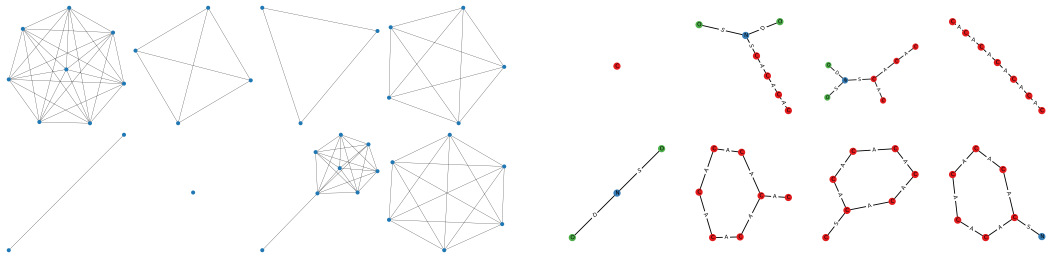

Figure 2: PnC + Neural Part. - Most probable graphs in the IMDB-B dataset (left) and the attributed MUTAG dataset (right). Atom types and bond types are represented as vertex and edge attributes.

7 Empirical Results

We evaluate our framework in a variety of datasets: small molecules, proteins and social networks [98–102]. Across all methods, we assume an optimal encoder that attains the entropy lower bound (this is often realistic since modern entropy coders asymptotically approach it) to evaluate the expressive power of each model independently of the encoder. We measure the description length of the data as their negative log-likelihood (NLL) under each probabilistic model, as well as the total description length by adding the cost of the parameters which need to be transmitted to the decoder (see Appendix D for details).

Baselines. We aim to assess representative approaches across the entire spectrum of graph probabilistic models, i.e., from completely uninformative non-parametric distributions to overparametrised neural generative models. We consider the following types of compressors: (a) *Null models.* We select the *uniform* model, where all edges are assumed to be sampled independently with probability equal to 0.5, the *edge list* model, a typical graph representation, and the *Erdős-Renyi* model (Eq. (2)). (b) *Partitioning-based.* non-parametric methods that aim at grouping vertices in tightly-connected clusters. They can be used for any type of sparse matrix [103] and are based on the assumption that there exists a hidden community structure in the graph. The partioning algorithms used are *SBM fitting* [44–47], Louvain [104] and Label Propagation [105] clustering. The encoding we use to encode the clusters corresponds exactly to the SBM assumptions, hence the partitioning-based results are always superior for this approach. (c) *Likelihood-based neural compressors.* As with any likelihood-based model, graph generative models can be transformed into graph compressors. We evaluate the original GraphRNN [66] and GRAN [68] networks, as well as smaller instantations that have undergone model compression using the Lottery Ticket Hypothesis algorithm [106].

Results. Tables 1 and 2 report the compression quality of each method measured in terms of the average number of bits required to store each edge in a dataset (bpe). We present four variants of PnC, differing on the type of partitioning algorithm used [105, 104, 47]. We report separately the cost of compressing the data as well as the total cost (including the parameters). Several observations can be made with regards to the baselines:

First off, *off-the-shelf likelihood-based neural approaches are poor compressors due to failing to address challenge C3.* These models exhibit an unfavorable trade-off between the data and model complexity, often requiring significantly more bpe than the null models. Although model compression techniques can alleviate this tradeoff (especially for larger datasets, e.g., pruned GraphRNN on ZINC), in most of the cases the compression ratios required to outperform PnC are significantly higher than the best that have been reported in the literature (See Table C.2.2 in the Appendix). Perhaps more importantly, it is unclear how to optimise the model description length during training (one of the few exceptions is [107]) and usually model compression might be tedious and is based on heuristics [108–111]. See Appendix C.2 for more details and additional experiments.

In addition, as expected from Theorem (1a), *non-parametric clustering algorithms work well when the dataset has a strong community structure, but are not a good choice for more structured datasets.* For instance, the best clustering algorithm requires 2.4× more bpe for ZINC than the best PnC. *PnC variants achieve the best compression in all datasets considered.* This follows from Theorem (1b), since the learned dictionaries are relatively small, and confirms our hypothesis that our framework is sufficiently flexible to account for the particularities of each dataset. As seen, neural partitioning

Table 1: Average bits per edge (bpe) for molecular graph datasets. **First**, **Second**, **Third**

Method type	Graph type	Small Molecules								
	Dataset name	MUTAG			PTC			ZINC		
		data	total	params	data	total	params	data	total	params
Null	Uniform (raw adjac.)	-	8.44	-	-	17.43	-	-	10.90	-
	Edge list	-	7.97	-	-	9.38	-	-	8.60	-
	Erdős-Renyi	-	4.78	-	-	5.67	-	-	5.15	-
Partitioning (non-parametric)	SBM-Bayes	-	4.62	-	-	5.12	-	-	4.75	-
	Louvain	-	4.80	-	-	5.27	-	-	4.77	-
	PropClust	-	4.92	-	-	5.40	-	-	4.85	-
Neural (likelihood)	GraphRNN	1.33	3338.21	388K	1.57	1394.59	389K	1.62	43,16	388K
	GRAN	0.81	12557.75	1460K	2.18	5269.82	1470K	1.30	157.7	1461K
	GraphRNN (pruned)	1.95	12.39	1.08K	2.16	6.71	1.10K	1.79	**2.02**	1.90K
	GRAN (pruned)	2.59	24.56	2.23K	4.31	14.00	2.36K	3.26	3.47	1.69K
PnC	PnC + SBM	3.81	4.11	49	4.38	4.79	155	3.34	3.45	594
	PnC + Louvain	2.20	**2.51**	47	2.68	**3.14**	166	1.96	**1.99**	196
	PnC + PropClust	2.42	**3.03**	63	3.38	**4.02**	178	2.20	2.35	726
	PnC + Neural Part.	2.17±0.02	**2.45±0.02**	46±1	2.63±0.26	**2.97±0.14**	143±31	2.01±0.02	**2.07±0.03**	384±105

Table 2: Average bits per edge (bpe) for social and protein graph datasets. **First**, **Second**, **Third**

Method type	Graph type	Biology			Social Networks					
	Dataset name	PROTEINS			IMDB-B			IMDB-M		
		data	total	params	data	total	params	data	total	params
Null	Uniform (raw adjac.)	-	24.71	-	-	2.52	-	-	1.83	-
	Edge list	-	10.92		-	8.29	-	-	7.74	-
	Erdős-Renyi	-	5.46	-	-	1.94	-	-	1.32	-
Partitioning (non-parametric)	SBM-Bayes	-	3.98	-	-	**0.80**	-	-	**0.60**	-
	Louvain	-	3.95	-	-	1.22	-	-	0.88	-
	PropClust	-	4.11	-	-	1.99	-	-	1.37	-
Neural (likelihood)	GraphRNN	2.03	156.99	392K	1.03	132.27	395K	0.72	127.84	392K
	GRAN	1.51	607.96	1545K	0.26	488.88	1473K	0.17	475.13	1467K
	GraphRNN (pruned)	2.63	3.76	2.56K	1.43	1.92	1.28K	0.91	1.39	1.28k
	GRAN (pruned)	4.28	5.11	1.78K	0.84	1.75	2.38K	0.55	1.41	2.31K
PnC	PnC + SBM	3.26	**3.60**	896	0.50	**0.54**	198	0.38	**0.38**	157
	PnC + Louvain	3.34	**3.58**	854	0.96	**1.02**	202	0.66	**0.70**	141
	PnC + PropClust	3.42	3.68	866	1.45	1.64	241	0.93	1.04	178
	PnC + Neural Part.	3.34±0.25	**3.51±0.23**	717±61	1.00±0.04	1.05±0.04	186±25	0.66±0.05	0.72±0.05	178±14

performs in every case better, or on par with the combination of PnC with the Louvain algorithm. However, in the social network datasets the combination of PnC with SBM achieves the best performance. This occurs because these networks fit nicely with the SBM inductive bias (which as a matter of fact is exactly that of low-entropy cuts), and most importantly, due to the fact that the clusters recovered by the SBM are small and repetitive, which makes them ideal for the PnC framework. We also observe that there is room for improvement for the neural partitioning variant, and hypothesise that a more powerful parametrised algorithm can be designed. Since learnable partitioning is still an open problem, we leave this research direction to future work.

Fig. 2 shows the most likely dictionary atoms for the IMDB-B and the MUTAG dataset (also including attributes - Appendix C.3 provides additional experiments). Observe that cliques or near-cliques and typical molecular substructures, such as carbon cycles and junctions are recovered for social networks and molecules respectively. This clearly highlights the connection between compression and pattern mining and provides evidence for potential applications of our framework.

8 Conclusion

This paper marks an important step towards learnable entropy-based graph compression. To the best of our knowledge, our work represents the first attempt to address the basic principles of parametric compressors of unlabelled graphs learned from observations. In addition, we suggest practical instantiations of our framework that can be trained with gradient-based optimisation, accounting for the total description length, hence aiming for the largest possible parsimony. A number of new research questions arise, such as how to design more expressive, albeit parsimonious distribution estimators, how to improve neural partitioning algorithms, and how to ensure scalability to single large networks that pose a significant challenge w.r.t the memory constraints of GPUs. We hope that our work will inspire further research in this emerging research area.

Acknowledgements and Disclosure of Funding

We would like to thank the anonymous reviewers for their valuable feedback and suggestions to improve our paper. This project was partially funded by the ERC Consolidator Grant No. 724228 - LEMAN. Giorgos Bouritsas is partially supported by a PhD scholarship from the Dpt. of Computing, Imperial College London. Andreas Loukas and Nikolaos Karalias thank the Swiss National Science Foundation for supporting them in the context of the project "Deep Learning for Graph Structured Data", grant number PZ00P2 179981. Michael Bronstein acknowledges support from Google Faculty awards and the Royal Society Wolfson Research Merit award.

References

[1] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[2] David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[3] László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 684–697, 2016.

[4] Paolo Boldi Sebastiano Vigna and P Boldi. The webgraph framework i: compression techniques. In *Proceedings of the 13th International Conference on World Wide Web*, pages 595–602, 2004.

[5] Yongsub Lim, U Kang, and Christos Faloutsos. Slashburn: Graph compression and mining beyond caveman communities. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3077–3089, 2014.

[6] Laxman Dhulipala, Igor Kabiljo, Brian Karrer, Giuseppe Ottaviano, Sergey Pupyrev, and Alon Shalita. Compressing graphs and indexes with recursive graph bisection. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1535–1544, 2016.

[7] Frank Harary and Edgar M Palmer. *Graphical enumeration*. Elsevier, 2014.

[8] Moni Naor. Succinct representation of general unlabeled graphs. *Discrete Applied Mathematics*, 28(3):303–307, 1990.

[9] Yongwook Choi and Wojciech Szpankowski. Compression of graphical structures: Fundamental limits, algorithms, and experiments. *IEEE Transactions on Information Theory*, 58(2):620–638, 2012.

[10] Jon Sneyers and Pieter Wuille. Flif: Free lossless image format based on maniac compression. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 66–70. IEEE, 2016.

[11] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1747–1756. JMLR.org, 2016.

[12] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10629–10638, 2019.

[13] Terry A Welch. A technique for high-performance data compression. *Computer*, 17(06):8–19, 1984.

[14] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.

[15] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE transactions on Information Theory*, 24(5):530–536, 1978.

[16] Timothy Bell, Ian H Witten, and John G Cleary. Modeling for text compression. *ACM Computing Surveys (CSUR)*, 21(4):557–591, 1989.

[17] Michael Burrows and David Wheeler. A block-sorting lossless data compression algorithm. In *Digital SRC Research Report*. Citeseer, 1994.

[18] Jürgen Schmidhuber and Stefan Heil. Sequential neural text compression. *IEEE Transactions on Neural Networks*, 7(1):142–146, 1996.

[19] Matthew V Mahoney. Fast text compression with neural networks. In *FLAIRS conference*, pages 230–234, 2000.

[20] Simon S. Du, Xiyu Zhai, Barnabás Póczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[21] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 244–253. PMLR, 2018.

[22] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 242–252. PMLR, 2019.

[23] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[24] Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

[25] Jarek Duda. Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding. *arXiv preprint arXiv:1311.2540*, 2013.

[26] Alberto Apostolico and Guido Drovandi. Graph compression by bfs. *Algorithms*, 2(3):1031–1044, 2009.

[27] Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *Proceedings of the 20th international conference on World Wide Web*, pages 587–596, 2011.

[28] Leonid Peshkin. Structure induction by lossless graph compression. In *2007 Data Compression Conference (DCC'07)*, pages 53–62. IEEE, 2007.

[29] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, Michael Mitzenmacher, Alessandro Panconesi, and Prabhakar Raghavan. On compressing social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 219–228, 2009.

[30] Francisco Claude and Gonzalo Navarro. Fast and compact web graph representations. *ACM Transactions on the Web (TWEB)*, 4(4):1–31, 2010.

[31] Ryan A Rossi and Rong Zhou. Graphzip: a clique-based sparse graph compression method. *Journal of Big Data*, 5(1):1–14, 2018.

[32] Sebastian Maneth and Fabian Peternek. Compressing graphs by grammars. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 109–120. IEEE, 2016.

[33] Maciej Besta and Torsten Hoefler. Survey and taxonomy of lossless graph compression and space-efficient graph representations. *arXiv preprint arXiv:1806.01799*, 2018.

[34] Ernesto Trucco. A note on the information content of graphs. *The bulletin of mathematical biophysics*, 18(2):129–135, 1956.

[35] Ernesto Trucco. On the information content of graphs: compound symbols; different states for each point. *The bulletin of mathematical biophysics*, 18(3):237–253, 1956.

[36] János Körner. Coding of an information source having ambiguous alphabet and the entropy of graphs. In *6th Prague conference on information theory*, pages 411–425, 1973.

[37] György Turán. On the succinct representation of graphs. *Discrete Applied Mathematics*, 8(3):289–294, 1984.

[38] Matthias Dehmer. Information-theoretic concepts for the analysis of complex networks. *Applied Artificial Intelligence*, 22(7-8):684–706, 2008.

[39] Matthias Dehmer, Nicola Barbarini, Kurt Varmuza, and Armin Graber. A large scale analysis of information-theoretic network complexity measures using chemical structures. *PLoS One*, 4(12):e8057, 2009.

[40] Abbe Mowshowitz and Matthias Dehmer. Entropy and the complexity of graphs revisited. *Entropy*, 14(3):559–570, 2012.

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

[42] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

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

[44] Tiago P Peixoto. Entropy of stochastic blockmodel ensembles. *Physical Review E*, 85(5):056122, 2012.

[45] Tiago P Peixoto. Parsimonious module inference in large networks. *Physical review letters*, 110(14):148701, 2013.

[46] Tiago P Peixoto. Nonparametric bayesian inference of the microcanonical stochastic block model. *Physical Review E*, 95(1):012317, 2017.

[47] Tiago P Peixoto. Bayesian stochastic blockmodeling. *Advances in network clustering and blockmodeling*, pages 289–332, 2019.

[48] Clement Lee and Darren J Wilkinson. A review of stochastic block models and extensions for graph clustering. *Applied Network Science*, 4(1):1–50, 2019.

[49] David Cox. Syntactically informed text compression with recurrent neural networks. *arXiv preprint arXiv:1608.02893*, 2016.

[50] Mohit Goyal, Kedar Tatwawadi, Shubham Chandak, and Idoia Ochoa. Deepzip: Lossless data compression using recurrent neural networks. *arXiv preprint arXiv:1811.08162*, 2018.

[51] James Townsend, Thomas Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[52] Friso H. Kingma, Pieter Abbeel, and Jonathan Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3408–3417. PMLR, 2019.

[53] James Townsend, Thomas Bird, Julius Kunze, and David Barber. Hilloc: lossless image compression with hierarchical latent variable models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[54] Yangjun Ruan, Karen Ullrich, Daniel Severo, James Townsend, Ashish Khisti, Arnaud Doucet, Alireza Makhzani, and Chris J. Maddison. Improving lossless compression rates via monte carlo bits-back coding. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 9136–9147. PMLR, 2021.

[55] Daniel Severo, James Townsend, Ashish Khisti, Alireza Makhzani, and Karen Ullrich. Compressing multisets with large alphabets. *arXiv preprint arXiv:2107.09202*, 2021.

[56] Chris S Wallace. Classification by minimum-message-length inference. In *International Conference on Computing and Information*, pages 72–81. Springer, 1990.

[57] Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

[58] Jonathan Ho, Evan Lohn, and Pieter Abbeel. Compression with flows via local bits-back coding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3874–3883, 2019.

[59] Emiel Hoogeboom, Jorn W. T. Peters, Rianne van den Berg, and Max Welling. Integer discrete flows and lossless compression. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12134–12144, 2019.

[60] Dustin Tran, Keyon Vafa, Kumar Krishna Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14692–14701, 2019.

[61] Rianne van den Berg, Alexey A. Gritsenko, Mostafa Dehghani, Casper Kaae Sønderby, and Tim Salimans. IDF++: analyzing and improving integer discrete flows for lossless compression. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[62] Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *arXiv preprint arXiv:2107.00630*, 2021.

[63] Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037*, 2021.

[64] Wengong Jin, Regina Barzilay, and Tommi S. Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *Proceedings of the 35th International Conference on Machine Learning, (ICML)*, volume 80, pages 2328–2337. PMLR, 2018.

[65] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[66] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5694–5703. PMLR, 2018.

[67] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L. Gaunt. Constrained graph variational autoencoders for molecule design. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 7806–7815, 2018.

[68] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard S. Zemel. Efficient graph generation with graph recurrent attention networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4257–4267, 2019.

[69] Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 2302–2312. PMLR, 2020.

[70] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. In *8th International Conference on Learning Representations, ICLR 2020*. OpenReview.net, 2020.

[71] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 7192–7203. PMLR, 2021.

[72] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[73] Carl Yang, Peiye Zhuang, Wenhan Shi, Alan Luu, and Pan Li. Conditional structure generation through graph variational generative adversarial nets. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 1338–1349, 2019.

[74] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In *Proceedings of the 35th International Conference on Machine Learning, (ICML)*, volume 80, pages 609–618. PMLR, 2018.

[75] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 4474–4484. PMLR, 2020.

[76] Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. Graphebm: Molecular graph generation with energy-based models. *arXiv preprint arXiv:2102.00546*, 2021.

[77] Thomas Bird, Friso H. Kingma, and David Barber. Reducing the computational cost of deep generative models with binary neural networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[78] Nikhil S Ketkar, Lawrence B Holder, and Diane J Cook. Subdue: Compression-based frequent pattern discovery in graph data. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, pages 71–76, 2005.

[79] Peter Bloem and Steven de Rooij. Large-scale network motif analysis using compression. *Data Mining and Knowledge Discovery*, 34(5):1421–1453, 2020.

[80] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. Vog: Summarizing and understanding large graphs. In *Proceedings of the 2014 SIAM international conference on data mining*, pages 91–99. SIAM, 2014.

[81] Kristen LeFevre and Evimaria Terzi. Grass: Graph structure summarization. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 454–465. SIAM, 2010.

[82] Farshad Nourbakhsh, Samuel Rota Bulo, and Marcello Pelillo. A matrix factorization approach to graph compression with partial information. *International Journal of Machine Learning and Cybernetics*, 6(4):523–536, 2015.

[83] Andreas Loukas and Pierre Vandergheynst. Spectrally approximating large graphs with smaller graphs. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3243–3252. PMLR, 2018.

[84] Andreas Loukas. Graph reduction with spectral and cut guarantees. *J. Mach. Learn. Res.*, 20(116):1–42, 2019.

[85] Vikas K. Garg and Tommi S. Jaakkola. Solving graph compression via optimal transport. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8012–8023, 2019.

[86] Yihe Dong and Will Sawin. Copt: Coordinated optimal transport on graphs. *Advances in Neural Information Processing Systems*, 33, 2020.

[87] Yu Jin, Andreas Loukas, and Joseph JaJa. Graph coarsening with preserved spectral properties. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 4452–4462. PMLR, 2020.

[88] Xuan Zhang, Xiaowen Dong, and Pascal Frossard. Learning of structured graph dictionaries. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3373–3376. IEEE, 2012.

[89] Cédric Vincent-Cuaz, Titouan Vayer, Rémi Flamary, Marco Corneli, and Nicolas Courty. Online graph dictionary learning. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 10564–10574. PMLR, 2021.

[90] Peter D Grünwald and Abhijit Grunwald. *The minimum description length principle*. MIT press, 2007.

[91] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[92] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[93] Bryan Wilder, Eric Ewing, Bistra Dilkina, and Milind Tambe. End to end learning and optimization on graphs. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4674–4685, 2019.

[94] Azade Nazi, Will Hang, Anna Goldie, Sujith Ravi, and Azalia Mirhoseini. Gap: Generalizable approximate graph partitioning framework. *arXiv preprint arXiv:1903.00614*, 2019.

[95] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 874–883. PMLR, 2020.

[96] Nikolaos Karalias and Andreas Loukas. Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[97] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[98] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374, 2015.

[99] John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768, 2012.

[100] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020.

[101] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005.

[102] Christoph Helma, Ross D. King, Stefan Kramer, and Ashwin Srinivasan. The predictive toxicology challenge 2000–2001. *Bioinformatics*, 17(1):107–108, 2001.

[103] Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S Modha, and Christos Faloutsos. Fully automatic cross-associations. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 79–88, 2004.

[104] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[105] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106, 2007.

[106] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[107] Marton Havasi, Robert Peharz, and José Miguel Hernández-Lobato. Minimal random code learning: Getting bits back from compressed model parameters. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[108] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[109] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3288–3298, 2017.

[110] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1135–1143, 2015.

[111] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.