One-Shot Compression of Large Edge-Exchangeable Graphs using Bits-Back Coding

Anonymous Authors[1]

Abstract

We present an optimal one-shot method for compressing large network graphs called *Bits-Back for Edge-Exchangeable Graphs*. The worst-case computational and memory complexities of our method scale quasi-linearly and linearly with the number of observed edges in the graph, making it efficient for compressing sparse real-world networks. Key to our method is bits-back coding, which is used to sample edges and vertices without replacement from the graph's edge-list in a way that preserves the structure of the network. Optimality is proven under a class of edge-exchangeable random graph models and experiments show the achieved bits-per-edge reaches the information content. We achieve competitive compression performance on real-world network datasets with millions of nodes and edges. The model used has only one parameter and can be learned via maximum likelihood with a simple line search.

1. Introduction

Network data, such as social networks and web graphs (Newman, 2018), can be represented as (large) graphs, or multigraphs, with millions of nodes and edges corresponding to members of a population and their interactions. Algorithms for compressing and storing network graphs must be *lossless*, as modifying edges can affect the underlying community structure as well as other relational aspects.

Compressing a single network is a *one-shot* lossless compression problem. This precludes the use of methods that require amortizing over repeated independent observations to reach the fundamental lower bound on lossless compression: the entropy (Cover, 1999). In the one-shot setting, the optimal number of bits to allocate for a single instance is equal to the negative log-likelihood (also known as the information content) under the estimated model.

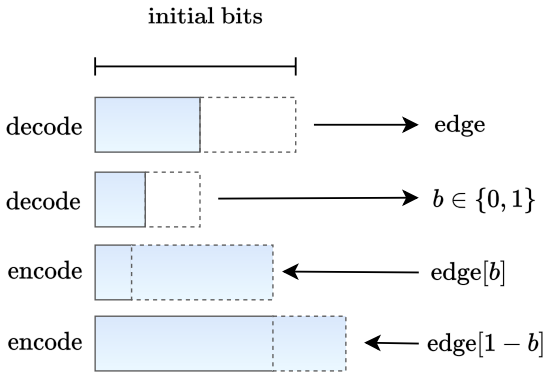

Figure 1: A diagram of our method, BB-EXG, compressing a single edge. The ANS stack is represented in blue. First, a random edge is sampled without replacement from the edge-list using an ANS decode. Then, a binary indicator b is decoded that defines the order in which vertices will be encoded: first edge$[b]$ then edge$[1-b]$.

Entropy coding is a popular lossless compression technique that makes use of a probabilistic model over the observed data. The main advantage of this approach over ad hoc methods is that it enables the use of existing domain knowledge on statistical modelling, which in the case of graphs is a well studied field (see Section 4.1). Bridging network modeling and data compression can yield significant gains in compression performance with little additional effort by the research community. For example, BB-ANS (Townsend et al., 2019) allowed powerful latent variable models to be easily repurposed for lossless data compression including variational autoencoders (Townsend et al., 2019), diffusion models (Kingma et al., 2021), and integer discrete flows (Hoogeboom et al., 2019; Berg et al., 2020), leading to state-of-the-art compression performance on image, speech (Havtorn et al., 2022) and smart meter time-series (Jeong et al., 2022) data. In many domains, current state-of-the-art compression algorithms such as CMIX (Knoll & de Freitas, 2012) and DeepZip (Goyal et al., 2019) use entropy coding together with a powerful neural network model.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

The entropy coding of a network can, in general, be done in quadratic time with respect to the number of vertices by storing binary variables indicating the absence or presence of each possible edge in the graph. However, most real-world networks are *sparse* in the sense that the number of edges is significantly lower than the maximum number of possible edges. Therefore, algorithms with sub-quadratic complexity in the number of observed edges are more attractive for compressing network data than those that scale quadratically with the number of nodes.

In this work we present the first optimal one-shot entropy coder for large graphs called *Bits-Back for Edge-Exchangeable Graphs* (BB-EXG). Our method is optimal under a broad class of distributions known as *finite edge-exchangeable* (Definition 5.1) and can achieve competitive performance on real-world networks as we show in Section 6. When paired with the model described in Section 5.2 the worst-case computational and memory complexities scale quasi-linearly and linearly with the number of observed edges in the graph.

BB-EXG uses *bits-back coding* (Frey & Hinton, 1996; Townsend et al., 2019) to sample edges and vertices without replacement from the graph's edge-list, similarly to (Severo et al., 2021a;b). Sampling is done by decoding from a shared random seed, which also stores the final message (i.e. the bits of the graph).

We define the notation used throughout the paper in Section 3. In Section 4.1 we discuss parameter-efficient models that yield good likelihood values on large sparse networks for which BB-EXG is optimal. We pair BB-EXG with a version of the edge-exchangeable *Hollywood model* (Crane & Dempsey, 2018), which we modify slightly to satisfy finite edge-exchangeability in Section 5.2, to achieve compression results competitive with the state-of-the-art on real-world network datasets in Section 6. The modified Hollywood model has only one parameter and can be learned via maximum likelihood via a simple line search.

2. Related work

To the best of our knowledge there is no previous entropy coding method that can scale to large graphs and is optimal for the broad class of finite edge-exchangeable graphs.

A number of previous works have presented adhoc methods for lossless compression of large graphs including Pool Compression (Yousuf & Kim, 2022), SlashBurn (Lim et al., 2014), List Merging (Grabowski & Bieniecki, 2014), BackLinks (Chierichetti et al., 2009), and Zuckerli (Versari et al., 2020). In sum, these methods attempt to exploit local statistics of the graph edge-list by defining an ordering of the vertex sequence that is amenable to compression. See Yousuf & Kim (2022) for an overview of the methods. Re-ordering techniques would yield no effect for finite edge-exchangeable models as all permutations of the vertex sequence have the same likelihood, as discussed in Section 5.1. In Section 6, Table 2 we compare the performance of these methods with that of entropy coding under the Hollywood model using our method and show that it performs competitively and can even outperform previous methods on sparser datasets.

Chen et al. (2021) develop a deep latent variable model where the edges are observations and the ordering of the vertices are latent. This could be made into a compression algorithm for a dataset of graphs by combining it with the entropy coder developed in (Townsend et al., 2019). It is unclear if this could be used for one-shot compression, where only a single graph is available. Furthermore, the authors present results on small graphs with a few hundred nodes and edges, while our setting is that of millions.

Another machine learning method for lossless graph compression is Partition and Code (Bouritsas et al., 2021). The method decomposes the graph into a collection of subgraphs and performs gradient descent to learn a dictionary code over subgraphs. While achieving good compression performance on small graph datasets, it is unclear if these methods can scale to networks with millions of nodes and edges.

Our work draws upon a large body of work on the statistical modeling of networks (Newman, 2018; Bloem-Reddy, 2017; Crane & Dempsey, 2018) which is reviewed in Section 4.1.

3. Notation

We use $[n]$ to represent a set of integers $\{1, \ldots, n\}$ and use superscript to represent sequences such as $x^n = x_1, \ldots, x_n$.

A graph sequence of n nodes, where an edge is added at each step, can be represented as a sequence of vertex elements v_i taking on values in the alphabet $[n]$ with the i-th edge defined as $e_i = (v_{2i-1}, v_{2i})$. We refer to the edge and vertex sequences interchangeably. The i-th graph is taken to be $G_i = \{\{v_1, v_2\}, \ldots, \{v_{2i-1}, v_{2i}\}\}$.

A *random graph model* is a mechanism that defines a joint distribution over vertex sequences either directly or through a sequence of conditionals.

Note that the likelihood, and hence the information content, of v^{2m} and e^m are the same, but they differ from that of G_m. A sequence of vertices or edges carries the information regarding the order in which the vertices and edges appeared in the graph sequence, which may be significantly larger than that of the graph.

We write $v \in v^{2m}, v \in e$ or $v \in G$ when the vertex v is present in the vertex sequence, edge, or graph and use $|v^i|$ to represent the number of unique vertices in v^i. The k-th

vertex of an edge is indicated by $e[k]$.

The ascending factorial function $a : \mathbb{R} \times \mathbb{N} \mapsto \mathbb{R}$ is defined by $a(x,k) = x(x+1)(x+2)\dots(x+k-1)$ and is abbreviated as $x^{\uparrow k}$.

All distributions are discrete, logarithms are base 2, and we refer to the probability mass function and cumulative distribution function as the PMF and CDF.

4. Background

4.1. Modeling real-world networks

Most real-world networks are *sparse* in the sense that the number of edges m is significantly smaller than the maximum number of possible edges $\binom{n}{2}$. For example, social networks are known to exhibit *small-world* characteristics (Newman, 2018) where most nodes are not connected by an edge but the degree of separation of any 2 nodes is small. In sparse graphs most interactions are local, i.e., a node will mostly interact with a small subset of neighbors. As the network grows over time, the maximum number of edges increases proportional to n^2, but the actual number of edges present in the graph grows sub-quadratically (Newman, 2018). A graph that is not sparse is known as a *dense* graph.

Random graph models that assign high likelihood to sparse graphs are thus preferred to model and compress real-world networks. A random graph model is called sparse if $\binom{n}{2}m^{-1} \to 0$ almost surely (i.e., with probability 1).

Vertex-exchangeable models, where the likelihood of a graph is invariant to relabeling of nodes, produce either dense or empty graphs with probability 1 (Aldous, 1981). An example is the Erdős–Rényi model $G(n,p)$ (Erdős et al., 1960) with p fixed. Each edge has a probability p of being present, which produces dense graphs as the expected number of edges is $p\binom{n}{2}$. This class includes other well known models such as the stochastic block model (Holland et al., 1983) and its mixed-membership variant (Airoldi et al., 2008). See Appendix B of (Cai et al., 2016) and (Lloyd et al., 2012) for a more complete list.

Edge-exchangeable models (Caron & Fox, 2017; Cai et al., 2016; Crane & Dempsey, 2018), where the likelihood of the edge-sequence is invariant to relabeling of the edges, achieve competitive likelihood values on network data (Crane & Dempsey, 2018). For this reason, they have been used to model network data in a number of applications including clustering (Sewell, 2020), anomaly detection (Luo et al., 2021), link-prediction (Williamson, 2016), community detection (Zhang & Dempsey, 2022), and have been extended to model hierarchical networks (Dempsey et al., 2021). The Hollywood model (Crane & Dempsey, 2018) discussed in Section 5.2 is an example of an edge-exchangeable model that can efficiently model network data. See the section for an in-depth discussion.

4.2. Lossless Compression with Entropy Coding

Given a discrete distribution P over alphabet $\mathcal{X}$, the objective of lossless compression is to find a code $C\colon \mathcal{X} \mapsto \{0,1\}^\star$ that minimizes the average code-length $\mathbb{E}_{x\sim P}\left[\ell(C(x))\right]$. Shannon's source coding theorem (Cover, 1999) guarantees that the average code-length is lower bounded by the *entropy* $H(P) = \mathbb{E}_{x\sim P}\left[-\log P(x)\right]$. A code with average code-length within 1 bit of the entropy is called *optimal* and assigns a code-word to $x \in \mathcal{X}$ with length close to its *information content* $\ell(C(x)) \approx -\log P(x)$.

Asymmetric Numeral Systems (ANS) is an algorithm that can implement an optimal code for $(\mathcal{X}, P)$ (Duda, 2009). ANS maps an instance $x \in \mathcal{X}$ to an integer state s via an encoding function. The instance can be losslessly recovered from the state via a decoding function which inverts the encoding procedure. Both encoding and decoding functions require access to the PMF and CDF as well as the current state. The last symbol encoded during compression is the first to be decoded during decompression implying ANS is *stack-like* (in contrast to most other entropy coders which are *queue-like*). For a more detailed discussion on ANS see (Townsend, 2020; 2021).

A key point to our method is that ANS can be used as a sampler by initializing the state to a random integer and performing successive decode operations with the distribution one wishes to sample from. This operation is invertible in the sense that the random seed can be fully recovered by performing an ANS encode with the sampled symbols. Decoding with distribution P reduces the size of the ANS state by approximately $\log 1/P(x)$ bits, where x is the sampled symbol, while encoding increases the state by the same amount. This is illustrated in Figure 1.

4.3. Bits-back coding with ANS

Bits-back coding with ANS (BB-ANS) is a general entropy coding method for latent variable models (Townsend et al., 2019). Given a model with observations x and latents z defined by an approximate posterior $Q(z \mid x)$, conditional likelihood $P(x \mid z)$ and prior $P(z)$, BB-ANS can perform tractable lossless compression of a sequence of observations.

In sum, BB-ANS projects the data x into the extended state-space (x,z) by sampling from $Q(z \mid x)$ while using the ANS state as a random seed and then encoding (x,z) with the joint distribution $P(x \mid z)P(z)$ (Ruan et al., 2021). The resulting ANS state is then used as a random seed to compress the next observation using the same procedure. Sampling with $Q(z \mid x)$ results in a reduction of the number of bits

in the ANS state. The average number of bits achieved by BB-ANS is a well known quantity called the (negative) *Evidence Lower Bound (ELBO)* which is an upper bound on the entropy of the data (Jordan et al., 1999). This method has been extended to hierarchical latent variable models (Kingma et al., 2019) and state space models (Townsend & Murray, 2021).

Our method can be understood under this framework as constructing a latent variable model with an exact approximate posterior, i.e., $Q(z \mid x) = P(z \mid x)$. The latent variable in our method is a Markov chain of sub-graphs of the graph being compressed.

The first step to encode a symbol is to decode a latent variable with the approximate posterior. As mentioned, this yields a savings of approximately $\log 1/Q(z \mid x)$ bits per observation. However, when encoding the first observation, the ANS stack is initially empty and therefore $\log 1/Q(z \mid x)$ bits must be encoded into the ANS state to allow for sampling. This initial overhead is known as the *initial bits problem* (Townsend et al., 2019) and in general is amortized by compressing multiple observations.

4.4. Entropy Coding for Large Alphabets

A large alphabet can render entropy coding intractable even if the PMF is tractable. For example, the alphabet size $|\mathcal{X}|$ of a graph grows exponentially with the number of nodes. Storing the PMF and CDF values in a hashtable provides fast look-ups but will in general require $\mathcal{O}(|\mathcal{X}|)$ memory which is impractical.

A common strategy to trade-off computational and memory complexity is to avoid compressing $(\mathcal{X}, P)$ directly and instead construct a bijection between $\mathcal{X}$ and an alphabet of proxy sequences y^m. For example, we can decompose a graph into a sequence of $\binom{n}{2}$ edge incidence variables each with alphabet sizes $|\mathcal{Y}_i| = 2$. Compression is then performed autoregressively with y_i conditioned on the history y^{i-1} and only the CDF and PMF of the current variable are stored in memory. The worst-case computational complexity is $\mathcal{O}\left(\sum_{i=1}^{m} c_i\right)$, where c_i represents the complexity of computing the PMF and CDF of y_i, while memory is $\mathcal{O}(1)$. If c_i's can be kept small then the trade-off can be made useful for practical applications.

Our method can be seen as an instance of this technique where we construct a bijection between the graph and a sequence of equivalence classes of vertex sub-sequences.

5. Method

In this section we present our entropy coder *Bits-Back for Edge-Exchangeable Graphs* (BB-EXG) which is, to the best of our knowledge, the first to perform optimal one-shot compression of a broad class of edge-exchangeable graphs.

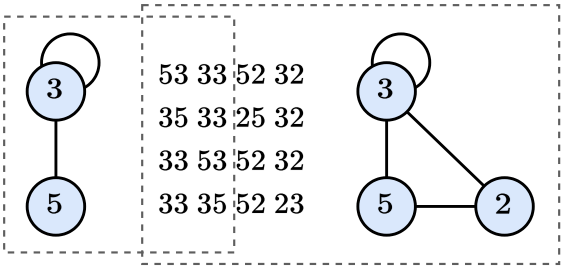

Figure 2: Example of the correspondence between vertex sequences and graph. Middle: 4 vertex sequences all corresponding to the same graphs. The vertices are labeled by single-digit integers and have been grouped to highlight the edge they belong to. Right: The graph corresponding to all 4 sequences. Left: The graph corresponding to the sub-sequences made up of the first 2 edges. In this particular case, all elements of the equivalence class of the graph are shown (i.e., all 4 sub sequences 53 33, 35 33, 33 53, and 33 35). For finite edge-exchangeable models (Definition 5.1), equivalent sequences are assigned the same likelihood.

Our method is optimal for any model over graphs satisfying finite edge-exchangeability (Definition 5.1). Section 5.1 discusses an example of such a model, which is parameterized by a single non-negative real number β, which is used in Section 6 to achieve competitive performance on network data. The model is based on the Hollywood model (Crane & Dempsey, 2018) which has been modified to satisfy finite edge-exchangeability.

See Section 3 for a review of the notation used.

5.1. Finite Edge-Exchangeability

BB-EXG is an optimal entropy coder for PMFs over vertex sequences that are invariant to permutations of the edges and of vertices within an edge. This is characterized formally by the following definition

Definition 5.1 (Finite Edge-Exchangeability)**.** Let v^{2m} be a vertex sequence with edges defined as $e_i = (v_{2i-1}, v_{2i})$ and σ an arbitrary permutation function over m elements. Given a collection $(\pi_k)_{k=1}^{m}$ of permutation functions over 2 elements, each over integers $(2k-1, 2k)$, we say that a model is *finite edge-exchangeable* if the following holds

$$\Pr(e_1, \ldots, e_m) = \Pr(\tilde{e}_{\sigma(1)}, \ldots, \tilde{e}_{\sigma(m)}), \tag{1}$$

where

$$\tilde{e}_k = (v_{\pi_k(2k-1)}, v_{\pi_k(2k)}). \tag{2}$$

The model we present next is invariant to permutations of vertices which is a sufficient, but not necessary, condition to satisfy finite edge exchangeability.

5.2. Finite Hollywood Model (FHM)

This section discusses a modified version of the Hollywood model (Crane & Dempsey, 2018), which has been adapted slightly to satisfy finite edge-exchangeability (Definition 5.1), and has been named the *Finite Hollywood model* (FHM).

The FHM defines a joint probability distribution over a vertex sequence v^k that is invariant to permutations of the elements in v^k. The joint PMF is defined via a sequence of conditional distributions

$$\Pr(v_{i+1} \,|\, v^i) \propto f_{v^i}(v_{i+1}) + \beta, \tag{3}$$

where $f_{v^i}(v) = \sum_{j=1}^{i} 1\{v = v_j\}$ is the frequency count of vertex v in v^i. Vertices that appeared more before have a higher likelihood of appearing again as the sequence grows.

The joint distribution can be expressed as

$$\Pr(v^k) = \frac{1}{(n\beta)^{\uparrow k}} \prod_{v \in U(v^k)} \beta^{\uparrow f_{v^k}(v)}, \tag{4}$$

where $U(v^k) = \{v \in [n] \colon v \in v^k\}$ is the set of unique vertices present in v^k and $x^{\uparrow k} = x(x+1)\dots(x+k-1)$ is the ascending factorial. The right-hand side is invariant to permutations of the vertices which is a sufficient, but not necessary, condition to satisfy Definition 5.1 implying the FHM can be used for optimal compression with BB-EXG. We provide a complete derivation in Appendix A.

The FHM can be seen as an instance of Polya's Urn model (Mahmoud, 2008) with each vertex corresponding to a different ball color. The urn is initialized with an equal number of balls for each of the n colours, corresponding to β (which need not be an integer). A ball (vertex) is sampled from the urn and then returned together with an extra ball of the same color. Well known properties of this model such as power-law behaviour and convergence are shared with the FHM. See (Mahmoud, 2008) for an in depth discussion.

Although the generative process is fairly simple the resulting joint distribution can achieve competitive likelihood values for sparse graphs as indicated in Table 2. Furthermore, we found empirically that fitting β via maximum likelihood is computationally tractable and, since β is the only parameter, can be done via line search. We showcase this optimization in Section 6 where we compress real-world networks with millions of nodes and edges.

5.3. BB-EXG: Bits-Back for Edge-Exchangeable Graphs

Compressing the graph with entropy coding requires computing the PMF and CDF of the graph from the PMF of the vertex sequence. Although the PMF of the graph for both the HM and FHM have closed form expressions, the alphabet size grows exponentially with the number of nodes which makes direct entropy coding infeasible (see Section 4.4).

We employ the technique discussed in Section 4.4 to trade-off computational and memory complexity by mapping the graph to a sequence of equivalence classes containing vertex sequences.

The vertex equivalence class $[v^{2m}]$ of a graph G with m edges is the set of all vertex sequences that map to G (see Figure 2 for an example). Models satisfying Definition 5.1 assign equal likelihood to sequences in the same equivalence class. Therefore, the negative loglikelihood of G, $[v^{2m}]$, and v^{2m} are related by

$$\log 1/\Pr(G) = \log 1/\Pr([v^{2m}]) \tag{5}$$

$$= \log 1/\Pr(w^{2m}) - \log|[w^{2m}]|, \tag{6}$$

for any equivalent w^{2m}, v^{2m} that map to graph G. The size of the equivalence class can be computed by counting the number of edge permutations and vertex permutations within an edge, which add up to

$$\log|[w^{2m}]| = m + \log m!. \tag{7}$$

The relationship between the likelihoods implies that we can reach the information content of the graph by compressing one of its vertex sequences if we can somehow get a number of bits back equal to $\log|[w^{2m}]|$. This leads to the naive Algorithm 1, which we describe below, that suffers from the initial bits issue (see Section 4.3).

Algorithm 1 Naive BB-EXG Encoder

Input: Vertex sequence v^{2m} and ANS state s
1) Edge-sort the vertex sequence v^{2m}
2) Decode a permutation σ uniformly w/ prob. $1/|[v^{2m}]|$
3) Apply the permutation to the vertex sequence
4) Encode the permuted vertex sequence
5) Encode m using $\log m$ bits

At step 1) we sort the vertex sequence without destroying the edge information by first sorting the vertices within an edge and then sorting the edges lexicographically. For example, edge-sorting sequence (34 12 32) yields (12 23 34). In step 2) an index is decoded that corresponds to a fixed permutation function agreed upon by the encoder and decoder, which is applied to the sequence in step 3). Note these permutations do not destroy the edge information by design. Finally, in step 4), the permuted sequence is encoded using $\Pr(v_{i+1} \,|\, v^i)$ from Section 5.2 followed by the number of edges. Decoding a permutation reduces the number of bits in the ANS state by exactly $\log|[v^{2m}]|$, while encoding the vertices increases it by $\log 1/\Pr(v^{2m})$. From (5), the net change is exactly the information content of the graph: $\log 1/\Pr(G)$.

The decoder acts in reverse order and perfectly inverts the encoding procedure, restoring the ANS state to its initial value. First, m is decoded. Then the sequence is decoded and the permutation is inferred by comparing it to its sorted version. Finally, the permutation is encoded to restore the ANS state.

Unfortunately, this method suffers from the initial bits problem (Townsend et al., 2019), as the decode step happens before encoding, implying there needs to be existing information in the ANS stack for the bit savings to occur. It is possible to circumvent this issue by incrementally sampling a permutation, similarly to (Severo et al., 2021a;b). This yields Algorithm 2, which we describe below, and is illustrated in Figure 1.

Algorithm 2 BB-EXG Encoder

Input: Vertex sequence v^{2m} and ANS state s.
1) Edge-sort the vertex sequence v^{2m}
repeat
  2) Decode an edge e_k uniformly from the sequence
  3) Remove e_k from the vertex sequence
  4) Decode a binary vertex-index b uniformly in $\{0, 1\}$
  5) Encode $e_k[b]$
  6) Encode $e_k[1-b]$
until The vertex sequence is empty
7) Encode m using $\log m$ bits.

BB-EXG progressively encodes the sequence by removing edges in a random order until the sequence is depleted. As before, we edge-sort the vertex sequence in step 1) without destroying the edge information. Then, in steps 2) and 3), an edge is sampled without replacement from the sequence by decoding an integer k between 1 and the size of the remaining sequence. Since the graph is undirected, we must destroy the information containing the order of the vertices in the edge. To do so, in step 4), we decode a binary index b and then encode vertices $e_k[b], e_k[1-b]$ in steps 5) and 6) using $\Pr(v_{i+1} \mid v^i)$. Finally, m is encoded.

The initial bits overhead is amortized as the number of edges grows. This makes BB-EXG an optimal entropy coder for large finite edge-exchangeable graphs, as characterized by the proposition below.

Proposition 5.2. *BB-EXG is an optimal entropy coder for any Finite Edge-Exchangeable Model (Definition 5.1) as* $m \to \infty$.

Proof. An optimal entropy coder compresses an object to within one bit of its information content (i.e., negative log-likelihood). Encoding steps add $\log 1/\Pr(v_i \mid v^{i+1})$ bits to the ANS state resulting in a total increase of $\log 1/\Pr(v^{2m})$ bits. Each decoding operation removes bits from the ANS state and together save $\sum_{i=1}^{m}(1+\log i) = \log 1/\Pr(|[v^{2m}]|)$. From (5), the net change is exactly the information content of the graph: $\log 1/\Pr(G)$. The initial and $\log m$ bits (needed to encode m) are amortized as $m \to \infty$. Therefore, the number of bits in the ANS state approaches (5), which concludes the proof. □

In Section 6 and Table 1 we show empirical evidence for the optimality of BB-EXG by compressing networks with millions of nodes and edges down to their information content under the Finite Hollywood Model.

5.4. BB-EXG for hypergraphs and directed graphs.

BB-EXG can be trivially extended to hypergraphs, where edges can have more than 2 nodes, directed graphs, and directed hypergraphs. The original Hollywood model (Crane & Dempsey, 2018) is defined for all 3 cases and the equivalent modifications done to yield the Finite Hollywood Model can also be performed.

For directed graphs, we need only ignore step 4) in Algorithm 2 and fix $b = 0$. This guarantees the order information between vertices in an edge is preserved.

For hypergraphs, steps 4-6) are generalized to the same sampling-without-replacement mechanism of step 3). After decoding an edge, the algorithm decodes vertices without replacement until the edge is depleted and encodes them in the order they appear. It then proceeds to decode another edge and repeats these 2 steps until the sequence is depleted.

5.5. Complexity Analysis

For a graph with m edges, the worst-case computational complexity of encoding and decoding with BB-EXG under the FHM is quasi-linear in the number of edges, $\mathcal{O}(m \log m)$, while the memory is linear: $\mathcal{O}(m)$.

We discuss only encoding with BB-EXG as decoding is analogous. The first step during encoding is to sort the edge-sequence which has worst-case complexity $\mathcal{O}(m \log m)$. Then, the edge-list is traversed and the frequency count of all vertices are stored in a binary search tree (BST) with at most $2m$ elements ($\mathcal{O}(m)$ memory). The BST allows for worst-case look-ups, insertions, and deletions in $\mathcal{O}(\log m)$, which are all necessary operations to construct $\Pr(v_{i+1} \mid v^i)$, as well as the CDF, used during entropy coding. Traversing the edge-list, together with the updates to the BST, require $\mathcal{O}(m \log m)$ computational complexity in the worst-case.

5.6. Pairing BB-EXG with arbitrary models.

While BB-EXG is only optimal for finite edge-exchangeable (FEXG) models, it can still be paired with any probability model over vertex sequences that have well defined conditional distributions such as (3).

For models that are not FEXG the order of the vertices will affect the likelihood assigned to the graph. BB-EXG in its current form will discount $m + \log m!$ bits, but all vertex sequences will have equal probability of appearing. The compression rate will therefore be

$$\log 1/\Pr(v^k) - (m + \log m!), \tag{8}$$

where v^k is the random sequence selected via the sampling-without-replacement mechanism of BB-EXG that maps to the compressed graph.

6. Experiments

In this section, we showcase the optimality of BB-EXG on large graphs representing sparse networks. The objective of this work is to develop an optimal entropy coder to enable tractable compression with a broad class of edge-exchangeable models. However, to illustrate the use of BB-EXG under a simple model, we entropy code with the FHM and compare the performance to state-of-the-art compression algorithms tailored to network compression. We report the average number of bits required to represent an edge in the graph (i.e., bits-per-edge) as is common in the literature.

We used datasets containing networks with small-world statistics (see Section 4.1) such as YouTube, FourSquare, Gowalla, and Digg (Rossi & Ahmed, 2015) which are expected to have high likelihood under the FHM. As negative examples, we compress Skitter and DBLP networks (Leskovec & Krevl, 2014), where we expect the results to be significantly worse than the state of the art, as these networks lack small-world statistics. The smallest network (Gowalla) has roughly 200 thousand nodes and 1 million edges, while the largest (Youtube) has more than 3 million nodes and almost 10 million edges.

The bias parameter β was fitted to minimize the negative log-likelihood (NLL) on each network via a search using SciPy's `scipy.optimize.minimize_scalar` (Virtanen et al., 2020). The log-likelihood can be computed efficiently by traversing the edge-list and calculating the likelihood under the FHM without performing compression.

We use the ANS implementation available in Craystack (Townsend et al., 2020; 2021). The cost of sending the bias parameter β as well as the number of edges m is negligible but is accounted for in the calculation of the BPE by adding 32 bits for each number (64 bits in total).

To compress a graph, the edges are loaded into memory as a list where each element is an edge represented by a tuple containing two vertex elements (integers). At each step, an edge is sampled without replacement using an ANS decode operation as described in Algorithm 2. Encoding is performed in a depth-first fashion, where an edge is encoded to completion before moving on to another. Then, a vertex is sampled without replacement from the edge and entropy-encoded using (3). The process repeats until the edge is depleted, and then starts again by sampling another edge without replacement. The process terminates once the edge-list is empty, concluding the encoding of the graph. Decoding is performed in reverse order and yields a vertex sequence that is equivalent (i.e., maps to the same graph as) the original graph.

Code implementing our entropy coder, the FHM, and experiments are available at `https://github.com/BB-EXG/bb-exg`.

6.1. Optimality of BB-EXG

We show empirical evidence for the optimality of BB-EXG by showing that BB-EXG can compress real-world graphs to the information content under the FHM (see Section 4.2).

Table 1 shows the negative loglikelihood (NLL) of the vertex sequence and graph under the Finite Hollywood model with β fitted via maximum likelihood. As discussed in Section 4.2 the graph's NLL is the value an optimal entropy coder should achieve to minimize the average number of bits with respect to the model. BB-EXG can compress the graph to its NLL (as indicated by the last column of Table 1) in all datasets except Gowalla. Compressing the graph as a sequence of vertices (i.e., without BB-EXG) would require a number of bits-per-edge equal to the sequence's NLL, which is significantly higher than the NLL of the graph as can be seen by the first column of Table 1.

These experiments indicate that BB-EXG can act as an optimal entropy coding method for finite edge-exchangeable models (Definition 5.1) which includes the modified version of the Hollywood model presented in Section 5.2. As these methods evolve to achieve better likelihood values the compression performance will improve automatically due to the optimality of BB-EXG.

6.2. Compressing Real-World Networks

In Table 2 we compare the bits-per-edge achieved by the Hollywood model using BB-EXG with current state-of-the-art algorithms for network data. The Finite Hollywood model performs competitively on all social networks and can even outperform previous works on sparser (lower density) networks such as YouTube (Rossi & Ahmed, 2015).

The likelihood assigned by the Hollywood model for non-social networks is expected to be low, resulting in poor compression performance, as indicated by the last 2 gray columns of Table 2. The performance of the Finite Hollywood model deteriorates as the edge density increases, as is expected (Crane & Dempsey, 2018), and is visible from Table 2.

Table 1: Optimality of BB-EXG with the Hollywood model. **BB-EXG achieves the optimal theoretical value for one-shot lossless compression under the Finite Hollywood model for various datasets.** Compressing the vertex sequence under the same model without BB-EXG would require a significantly higher number of bits-per-edge as indicated by the sequence's negative log-likelihood (NLL). All units are in bits-per-edge.

NETWORK	SEQ. NLL	BB-EXG (OURS)	GRAPH NLL (OPTIMAL)	GAP (%)
YOUTUBE	37.73	15.01	15.01	0.0
TWITTER	36.82	16.14	16.14	0.0
SKITTER	37.18	14.22	14.22	0.0
DBLP	35.45	15.89	15.89	0.0
FOURSQUARE	30.99	9.82	9.82	0.0
GOWALLA	32.10	12.18	11.68	4.3

Table 2: Lossless compression of real-world networks. The Finite Hollywood model (FHM), together with BB-EXG, achieves competitive performance on some datasets and can even outperforms the current state-of-the-art on sparser social networks (black columns to the left). **The last 2 columns (in gray) highlight the situations where the FHM is expected to under-perform.** While the compression with BB-EXG is optimal, the final results depend on the likelihood assigned to the graph under the FHM, which is why ad hoc methods can achieve a better performance. The best results are highlighted in bold. Results for methods beyond ours are the ones reported by (Yousuf & Kim, 2022) in Table 4. Units for the bottom section are in bits-per-edge and **lower numbers indicate better performance.**

	SOCIAL NETWORKS				OTHERS	
	YOUTUBE	FOURSQ.	DIGG	GOWALLA	SKITTER	DBLP
# NODES	3,223,585	639,014	770,799	196,591	1,696,415	317,080
# EDGES	9,375,374	3,214,986	5,907,132	950,327	11,095,298	1,049,866
$10^6 \times$DENSITY	1.8	15.8	19.8	50.2	7.7	20.9
OPTIMAL β	1.32	0.71	0.54	1.01	1.08	2.51
(OURS) FHW W/ BB-EXG	**15.01**	9.82	10.49	12.18	14.22	15.89
POOL COMP.	15.38	**9.23**	11.59	**11.73**	7.45	**8.78**
SLASHBURN	17.03	10.67	**9.82**	11.83	12.75	12.62
BACKLINKS	17.98	11.69	12.56	15.56	11.49	10.79
LIST MERGING	15.80	9.95	11.92	14.88	**8.87**	14.13

While the compression with BB-EXG is optimal for the FHM, the final results depend on the likelihood assigned to the graph under the FHM, which is why ad hoc methods can achieve better performance. Nonetheless, the bits-per-edge of FHM with BB-EXG is close to that of current methods.

The graph NLL for the Gowalla network under the FHM shown in Table 1 (11.68) is less than the best compression result (11.73) achieved by Pool Compression (Yousuf & Kim, 2022). Assuming the network will grow with similar statistics, the FHM may eventually surpass Pool Compression as there would be more edges to amortize the overheads in compression.

7. Conclusion

In this paper we developed the first algorithm capable of performing tractable one-shot entropy coding of large finite edge-exchangeable graphs: *Bits-Back for Edge-Exchangeable Graphs* (BB-EXG). We provide an example use case using a modified version of the self-reinforcing Hollywood model (Crane & Dempsey, 2018) which performs competitively with state of the art methods despite having only 1 parameter (the bias β). Optimality of BB-EXG is proven for finite edge-exchangeable models and demonstrated empirically on a variety of real-world networks. This optimality implies that the efforts from the graph modeling community in improving the likelihood of network data under edge-exchangeable models can be directly translated into an increased performance in lossless compression tasks, as BB-EXG can compress graphs to the theoretical optimum (negative log-likelihood).

Our method can be seen as an extension of the Bits-Back with ANS method (Townsend et al., 2019) that is applicable to one-shot compression, i.e., when only a single sample from the data distribution is available. BB-EXG can be trivially extended to hypergraphs and directed edges as men-

tioned in Section 5.4.

Any model satisfying finite edge-exchangeability (Definition 5.1) can be used for optimal compression with BB-EXG, including neural network based models. Learning exchangeable models has been explored in the literature (Niepert & Domingos, 2014) but, to the best of our knowledge, using them for compression of graphs and other structured data is an under-explored field.

The FHM presented in this paper satisfies finite edge-exchangeability through the invariance of the PDF to permutations of the vertices, which is a sufficient, but not necessary, condition. An interesting direction to investigate is if there are similar models that are strictly finite edge-exchangeable, that is, the PDF is invariant to permutations of edges and vertices within an edge, but not to permutation of vertices from different edges. Appendix B discusses these directions further.

In general, a trade-off exists between the model performance and the complexity required to compute the conditional distributions. The FHM lies on an attractive point of this trade-off curve, but there might exist other methods that perform better without increasing complexity significantly. We think this is a promising line of work that can yield better likelihood models for network data and can provide a principled approach to lossless compression of these data types.

References

Airoldi, E. M., Blei, D., Fienberg, S., and Xing, E. Mixed membership stochastic blockmodels. *Advances in neural information processing systems*, 21, 2008.

Aldous, D. J. Representations for partially exchangeable arrays of random variables. *Journal of Multivariate Analysis*, 11(4):581–598, 1981.

Berg, R. v. d., Gritsenko, A. A., Dehghani, M., Sønderby, C. K., and Salimans, T. Idf++: Analyzing and improving integer discrete flows for lossless compression. *arXiv preprint arXiv:2006.12459*, 2020.

Bloem-Reddy, B. M. *Random Walk Models, Preferential Attachment, and Sequential Monte Carlo Methods for Analysis of Network Data*. PhD thesis, Columbia University, 2017.

Bouritsas, G., Loukas, A., Karalias, N., and Bronstein, M. Partition and code: learning how to compress graphs. *Advances in Neural Information Processing Systems*, 34: 18603–18619, 2021.

Cai, D., Campbell, T., and Broderick, T. Edge-exchangeable graphs and sparsity. *Advances in Neural Information Processing Systems*, 29, 2016.

Caron, F. and Fox, E. B. Sparse graphs using exchangeable random measures. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 79(5):1295, 2017.

Chen, X., Han, X., Hu, J., Ruiz, F. J., and Liu, L. Order matters: Probabilistic modeling of node sequence for graph generation. *arXiv preprint arXiv:2106.06189*, 2021.

Chierichetti, F., Kumar, R., Lattanzi, S., Mitzenmacher, M., Panconesi, A., and Raghavan, P. On compressing social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 219–228, 2009.

Cover, T. M. *Elements of information theory*. John Wiley & Sons, 1999.

Crane, H. and Dempsey, W. Edge exchangeable models for interaction networks. *Journal of the American Statistical Association*, 113(523):1311–1326, 2018.

Dempsey, W., Oselio, B., and Hero, A. Hierarchical network models for exchangeable structured interaction processes. *Journal of the American Statistical Association*, pp. 1–18, 2021.

Duda, J. Asymmetric numeral systems. *arXiv:0902.0271 [cs, math]*, 2009.

Erdős, P., Rényi, A., et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

Frey, B. J. and Hinton, G. E. Free energy coding. In *Proceedings of Data Compression Conference-DCC'96*, pp. 73–81. IEEE, 1996.

Goyal, M., Tatwawadi, K., Chandak, S., and Ochoa, I. Deepzip: Lossless data compression using recurrent neural networks. In Bilgin, A., Storer, J., Marcellin, M., and Serra-Sagrista, J. (eds.), *Proceedings - DCC 2019*, Data Compression Conference Proceedings, United States, 5 2019. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/DCC.2019.00087.

Grabowski, S. and Bieniecki, W. Tight and simple web graph compression for forward and reverse neighbor queries. *Discrete Applied Mathematics*, 163:298–306, 2014.

Hartford, J., Graham, D., Leyton-Brown, K., and Ravanbakhsh, S. Deep models of interactions across sets. In *International Conference on Machine Learning*, pp. 1909–1918. PMLR, 2018.

Havtorn, J. D., Borgholt, L., Hauberg, S., Frellsen, J., and Maaløe, L. Benchmarking generative latent variable models for speech. *arXiv preprint arXiv:2202.12707*, 2022.

Holland, P. W., Laskey, K. B., and Leinhardt, S. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

Hoogeboom, E., Peters, J., Van Den Berg, R., and Welling, M. Integer discrete flows and lossless compression. *Advances in Neural Information Processing Systems*, 32, 2019.

Jeong, H., Seo, G., and Hwang, E. Lossless data compression with bit-back coding on massive smart meter data. In *2022 IEEE International Conference on Big Data (Big Data)*, pp. 6667–6669. IEEE, 2022.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine learning*, 37:183–233, 1999.

Kingma, D., Salimans, T., Poole, B., and Ho, J. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Kingma, F., Abbeel, P., and Ho, J. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. In *International Conference on Machine Learning*, pp. 3408–3417. PMLR, 2019.

Knoll, B. and de Freitas, N. A machine learning perspective on predictive coding with paq8. In *2012 Data Compression Conference*, pp. 377–386. IEEE, 2012.

Leskovec, J. and Krevl, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

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

Lloyd, J., Orbanz, P., Ghahramani, Z., and Roy, D. M. Random function priors for exchangeable arrays with applications to graphs and relational data. *Advances in Neural Information Processing Systems*, 25, 2012.

Luo, R., Nettasinghe, B., and Krishnamurthy, V. Anomalous edge detection in edge exchangeable social network models. *arXiv preprint arXiv:2109.12727*, 2021.

Mahmoud, H. *Pólya urn models*. CRC press, 2008.

Meng, C., Yang, J., Ribeiro, B., and Neville, J. Hats: A hierarchical sequence-attention framework for inductive set-of-sets embeddings. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 783–792, 2019.

Newman, M. *Networks*. Oxford university press, 2018.

Niepert, M. and Domingos, P. Exchangeable variable models. In *International Conference on Machine Learning*, pp. 271–279. PMLR, 2014.

Rossi, R. and Ahmed, N. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

Ruan, Y., Ullrich, K., Severo, D., Townsend, J., Khisti, A., Doucet, A., Makhzani, A., and Maddison, C. J. Improving Lossless Compression Rates via Monte Carlo Bits-Back Coding. In *International Conference on Machine Learning*, 2021.

Severo, D., Townsend, J., Khisti, A., Makhzani, A., and Ullrich, K. Compressing multisets with large alphabets, 2021a.

Severo, D., Townsend, J., Khisti, A. J., Makhzani, A., and Ullrich, K. Your dataset is a multiset and you should compress it like one. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021b. URL https://openreview.net/forum?id=vjrsNCu8Km.

Sewell, D. K. Model-based edge clustering. *Journal of Computational and Graphical Statistics*, 30(2):390–405, 2020.

Townsend, J. A tutorial on the range variant of asymmetric numeral systems. *arXiv:2001.09186 [cs, math, stat]*, 2020.

Townsend, J. Lossless compression with latent variable models. *arXiv preprint arXiv:2104.10544*, 2021.

Townsend, J. and Murray, I. Lossless compression with state space models using bits back coding. In *Neural Compression: From Information Theory to Applications – Workshop at ICLR*, 2021.

Townsend, J., Bird, T., and Barber, D. Practical lossless compression with latent variables using bits back coding. In *International Conference on Learning Representations (ICLR)*, 2019.

Townsend, J., Bird, T., Kunze, J., and Barber, D. Hi{ll}oc: lossless image compression with hierarchical latent variable models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1lZgyBYwS.

Townsend, J., Bird, T., Kunze, J., Severo, D., and Conzel. j-towns/craystack:, August 2021. URL https://doi.org/10.5281/zenodo.5180977.

Versari, L., Comsa, I.-M., Conte, A., and Grossi, R. Zuckerli: A new compressed representation for graphs. *IEEE Access*, 8:219233–219243, 2020.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Williamson, S. A. Nonparametric network models for link prediction. *Journal of Machine Learning Research*, 17(202):1–21, 2016. URL http://jmlr.org/papers/v17/16-032.html.

Yousuf, M. I. and Kim, S. Pool compression for undirected graphs. *IEEE Access*, 2022.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Zhang, Y. and Dempsey, W. Node-level community detection within edge exchangeable models for interaction processes. *arXiv preprint arXiv:2208.08539*, 2022.

A. Proving the Finite Hollywood Model is Finite Edge-Exchangeable (Definition 5.1).

In this section we prove the Finite Hollywood Model is finite edge exchangeable and thus can be used for optimal compression with BB-EXG. The proof follows from directly computing the joint $\Pr(v^k)$ from the conditionals $\Pr(v_i \mid v^{i-1})$ and showing that it depends on factors that are invariant to permutations of the vertices in v^k.

The conditional is proportional to the frequency count of previous vertices and is biased by the parameter β,

$$\Pr(v_{i+1} \mid v^i) \propto f_{v^i}(v_{i+1}) + \beta, \tag{9}$$

with normalizing constant

$$\sum_{v_{i+1} \in [n]} (f_{v^i}(v_{i+1}) + \beta) = i + n\beta. \tag{10}$$

The joint is defined as the product of the conditionals,

$$\Pr(v^k) = \prod_{i=0}^{k-1} \frac{f_{v^i}(v_{i+1}) + \beta}{i + n\beta}. \tag{11}$$

At step i, the generative process appends a vertex to the existing sequence resulting in the product of non-decreasing frequency counts (plus the bias β) in the numerator. We can regroup the frequency terms and rewrite it as a function of the final counts f_{v^k}. For example, consider the following sequence and its joint distribution

$$v^k = 12\ 23\ 21 \tag{12}$$

$$\Pr(v^k) = \frac{\overbrace{\beta}^{v_1=1}}{n\beta} \cdot \frac{\overbrace{\beta}^{v_2=2}}{1+n\beta} \cdot \frac{\overbrace{1+\beta}^{v_3=2}}{2+n\beta} \cdot \frac{\overbrace{\beta}^{v_4=3}}{3+n\beta} \cdot \frac{\overbrace{2+\beta}^{v_5=2}}{4+n\beta} \cdot \frac{\overbrace{1+\beta}^{v_6=1}}{5+n\beta} \tag{13}$$

$$= \frac{1}{\prod_{i=0}^{k-1}(i+n\beta)} \cdot \underbrace{\beta \cdot (1+\beta)}_{v_1=v_6=1} \cdot \underbrace{\beta \cdot (1+\beta) \cdot (2+\beta)}_{v_2=v_3=v_5=2} \cdot \underbrace{\beta}_{v_4=3}. \tag{14}$$

In general, the joint takes on the form

$$\Pr(v^k) = \frac{1}{\prod_{i=0}^{k-1}(i+n\beta)} \prod_{v \in U(v^k)} (\beta)(1+\beta)\dots(f_{v^k}(v) - 1 + \beta) \tag{15}$$

$$= \frac{1}{(n\beta)^{\uparrow k}} \prod_{v \in U(v^k)} \beta^{\uparrow f_{v^k}(v)}, \tag{16}$$

$$\tag{17}$$

where $U(v^k) = \{v \in [n] \colon v \in v^k\}$ is the set of unique vertices present in v^k and $x^{\uparrow k} = x(x+1)\dots(x+k-1)$ is the ascending factorial. The expression on the right is clearly invariant to permutations of the elements in v^k, which concludes the proof.

B. Other Finite Edge-Exchangeable Models.

B.1. Extended Finite Hollywood Model

Note that assigning a unique bias β_v to each vertex v will not break finite edge-exchangeability of the Finite Hollywood Model, as can be seen from

$$\Pr(v_{i+1} \mid v^i) = \frac{f_{v^i}(v_{i+1}) + \beta_{v_{i+1}}}{i + \sum_{v \in [n]} \beta_v}, \tag{18}$$

with corresponding joint distribution

$$\Pr(v^k) = \frac{1}{(n\sum_{v\in[n]}\beta_v)^{\uparrow k}} \prod_{v\in U(v^k)} (\beta_v)^{\uparrow f_{v^k}(v)}. \tag{19}$$

$$\tag{20}$$

The parameters $\{\beta_v\}_{v\in[n]}$ can be learned via gradient descent methods as the gradient of the joint is easily computable. However, this model has n parameters, one for each vertex, which would need to be transmitted together with the model depending on how the model generalizes as the network grows. We did not explore this direction.

B.2. Strictly Finite Edge-Exchangeable Models via nested Deep Sets (Zaheer et al., 2017).

In this section we outline a general framework based on (Zaheer et al., 2017; Hartford et al., 2018; Meng et al., 2019) to construct joint distributions that are finite edge-exchangeable (FEXG), but are not invariant to permutations of vertices between different edges. We say these models are *strictly* FEXG.

Let $\psi\colon [n]^2 \mapsto \mathbb{R}^\ell$ and $\Psi\colon \bigcup_{k\in\mathbb{N}}(\mathbb{R}^\ell)^k \mapsto \mathbb{R}^+$ be functions invariant to permutation of their arguments, possibly parameterized by some neural network. The following joint distribution is clearly FEXG,

$$\Pr(v^{2m}) \propto \Psi(\psi(v_1, v_2), \psi(v_3, v_4), \ldots, \psi(v_{2m-1}, v_{2m})). \tag{21}$$

However, different from the FHM, the joint distribution may not be invariant to permutations of vertices between different edges (as intended, making it strictly FEXG). As a concrete example, take

$$\psi(v, w) = \theta_v + \theta_w, \tag{22}$$

$$\Psi(\phi_1, \ldots, \phi_{2m}) = \langle \theta_v, \theta_w \rangle. \tag{23}$$

where $\theta_v, \theta_w \in \mathbb{R}^\ell$ are embeddings that can be learned and $\langle\cdot,\cdot\rangle$ represents an inner-product.

To apply BBEXG, we need to define the conditional distributions

$$\Pr(v_{2i}, v_{2i-1} \,|\, v^{2(i-1)}) = \frac{\Pr(v^{2i})}{\sum_{v_{2i}, v_{2i-1}} \Pr(v^{2i})}. \tag{24}$$

This model can also be learned via stochastic gradient descent but quickly becomes intractable in the form presented for graphs with millions of edges.