Partition and Code: Learning How to Compress Graphs

(Bouritsas et al. 2021)

Yuen Chen, Shane Wang, Peizhi Niu

October 29th, 2024

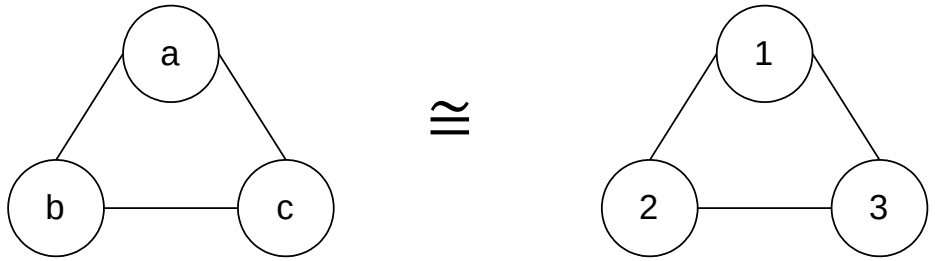
Motivations:

- ***Graph data become more prevalent:***
  - Abundance of web-graphs, social networks, and biological networks.

- ***Limitations of conventional compression algorithms for graph data:***
  - Unlike text, images or videos, graphs lack inherent ordering.

- ***Main Challenges in Graph Compression:***
  C1. Dealing with Graph Isomorphism (GI)
  C2. Evaluating the Likelihood
  C3. Accounting for the Description Length of the Learned Model

Challenges in Graph Compression



C1. Dealing with Graph Isomorphism

- ***Graph Isomorphism*** *(Informal):* two graphs are isomorphic if they have the same structure but potentially different vertex labels.
- To avoid redundant storage, isomorphic graphs should be encoded with the same codeword.
- Unfortunately, the complexity of identifying graph isomorphism is super-polynomial on the number of vertices.
- ***Current approaches:***
  - Optimize an ordering of the vertex and encode graphs with vector-based methods.
  - $\Longrightarrow$ Loss in compression

Challenges in Graph Compression

C2. Evaluating the Likelihood

- An optimal encoder need accurate estimate of the probability of possible outcomes (to assign less bits to more probable ones.)
- In high dimensional data, this is usually done by partitioning the data into smaller parts.
  - Image data: distribution of pixels/patches.
  - Text data: distribution of characters or n-grams.

- ***Issue with graph data:*** Unclear how to decompose the data without canonical ordering.

Challenges in Graph Compression

C3. Accounting for the Description Length of the Learned Model

- Deep learning models data distribution with overparametrized neural networks for better generalization
- In compression, the encoder, i.e., parameters of the neural network, needs to be stored for decoding
  - Large description length for overparametrized model
  - $\Longrightarrow$ Inefficient

Related Work

Engineered Codes

- Majority of the graph compression methods are NOT probabilistic.
  - Rely on hand-engineered encodings optimized to incorporate domain knowledge.
- Ideas:
  - Reorder the vertices (permute the adjacency matrix) such that the graph is compression friendly for sequence compressors (Abbe, 2016).
  - Identify frequent substructures for more efficient representation.

- Often based on heuristics such as specific network properties.
- Do not model the underlying graph distribution.

Related Work

Theory-driven approaches

- Few works have attempted to model the graph distribution
  - ***Stochastic Block Models:*** generate graphs with community structure
- Any graph clustering algorithm can be used for compression
  - (Empirically) But are less effective

Related Work

Likelihood-based neural approaches

- Use generative models for compression:
  - Autoregressive models
  - Variational autoencoders
  - Normalizing flows
  - Diffusion models
- Drawbacks:
  - Compute probability on labeled graphs instead of isomorphic class (*C1*)
  - Use heuristic ordering
  - Parameter inefficient

Preliminaries

Notations

- $G = \{V, E\}$ is a **graph** with
  - n vertices $V = \{v_1, \dots, v_n\}$ and
  - m edges $E = \{e_{i,j} \mid s.t.\ v_i,\ v_j \in V\ are\ connected\ by\ an\ edge\}$
- Observation Space: $\mathfrak{G}$
- Samples: $\mathcal{G} = \{G_1, \dots, G_{|\mathcal{G}|}\} \subset \mathfrak{G}$
- Description method/symbol code: $CODE : \mathfrak{G} \to \{0,1\}^*$
  - *Input:* A graph
  - *Output:* a variable-length sequence of binary symbol (codeword)

Preliminaries

Information Theory

- *Goal:* Minimize description length of the code, $L_{CODE}(G)$
  - i.e., number of bits needed to encode G
  - The codes are uniquely decodable
- Kraft-McMillan inequality implies probability distribution:

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

  - The entropy $H_q[G]$ is a lower bound on the average codelength
  - $\Longrightarrow$ compression is equivalent to define a distribution with smallest entropy

Preliminaries

Baseline Graph Encoding

- Uniform random graph model:
  - Assign equal probability all labeled graph with n vertices and m edges
  - Encoding length (assume uniform probability of n and $m|\mathrm{n}$):

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

Encode # of vertices

Encode # of edges

Encode # of ways to arrange m edges

Methodologies

3 modules:

- Partitioning: decompose into subgraphs
- Graph dictionary: collection of subgraphs
- Graph encoding: transfer subgraphs to bits

Methodologies

Partitioning:

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

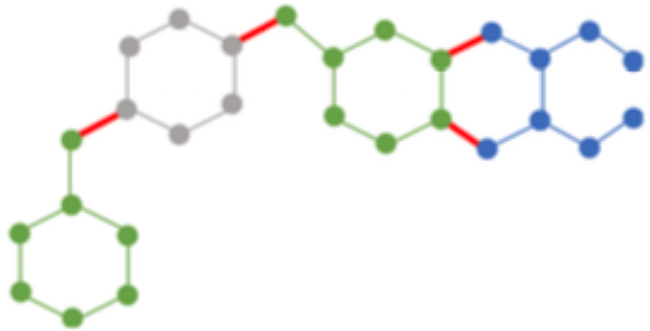


Figure 1: Illustration of the graph decomposition. The subgraph colours correspond to dictionary atoms a_1, a_2 and a_3. Cuts are denoted in red.

$H_i = \{V_i, E_i\}$ is the i-th subgraph

$C = \{V, E_C\}$ is a bipartite graph containing all cut edges $E_C = E - U_i E_i$

$$p_\theta(S_T|G) = p_\theta(H_T|S_{T-1}^H, G)p_\theta(S_{T-1}^H|G) = \prod_{t=1}^{T} p_\theta(H_t|S_{t-1}^H, G)$$

Vertex selection probability — Vertex count probability

$$= \prod_{t=1}^{T} \left(\prod_{i=1}^{k_t} p_\theta(v|S_{i-1}^V, k_t, S_{t-1}^H, G) \right) p_\theta(k_t|S_{t-1}^H, G)$$

We only need to parameterize this two probabilities

$p_\theta(S_T)$: probability of S_T

$S_t^H = \{H_1, H_2, \ldots H_t\}$: subgraph state (decisions made at the subgraph level up to step t)

$S_i^V = \{v_{t1}, v_{t2}, \ldots v_{ti}\}$: vertex state (decisions made at the vertex level up to i-th vertex selection)

T: number of iterations

Deepset: https://arxiv.org/abs/1703.06114

Methodologies

Partitioning:

GNN embedding

$$\mathbf{h}(v) = \mathrm{GNN}_v(G) \quad \mathbf{h}(G) = \mathrm{GNN}_G(G)$$

Set function approximator

$$\mathbf{h}(H_t) = \mathrm{DeepSets}(\{\mathbf{h}(v) | v \in H_t\}) \quad \mathbf{h}(S_t^H) = \mathrm{DeepSets}(\{\mathbf{h}(H_t) | H_t \in S_t^H\})$$

Algorithm 1: Partitioning algorithm

Input: graph G
Output: partition $\mathcal{H}$
Initialisations: $\mathbf{h}(v) = \mathrm{GNN}_v(G)$, $\mathbf{h}(G) = \mathrm{GNN}_G(G)$, $V_1 = V$, $S_0^H = \emptyset$
$t \leftarrow 1$
while $V_t \neq \emptyset$ **do**
  $k_t \sim p_\theta(k_t | S_{t-1}^H, G)$ // sample maximum vertex count
  Initialise $S_0^V = \emptyset$
  while $i = 1 \leq k_t$ *and* $\mathcal{N}(S_{i-1}^V) \neq \emptyset$ **do**
    $v_{t_i} \sim p_\theta(v | S_{i-1}^V, k_t, S_{t-1}^H, G)$ // sample new vertex
    $S_i^V = S_{i-1}^V \cup \{v_{t_i}\}$
  end
  $H_t = S_i^V$
  $S_t^H = S_{t-1}^H \cup \{H_t\}$
  $t \leftarrow t + 1$
end
$\mathcal{H} = S_t^H$

$$p_\theta(k_t | S_{t-1}^H, G) = \mathrm{softmax}_{k_t=1}^{|V_t|} \mathrm{MLP}\big(\mathbf{h}(S_{t-1}^H), \mathbf{h}(G)\big)$$

Vertex count probability

$$p_\theta(v | S_{i-1}^V, k_t, S_{t-1}^H, G) = \begin{cases} \mathrm{softmax}_{v \in V_t} \mathrm{MLP}\big(\mathbf{h}(S_{t-1}^H), \mathbf{h}(v)\big) & i = 0, v \in V_t \\ \mathrm{softmax}_{v \in V_t \cap \mathcal{N}(S_{i-1}^V)} \mathrm{MLP}\big(\mathbf{h}(S_{t-1}^H), \mathbf{h}(v)\big) & 0 < i < k_t, v \in V_t \cap \mathcal{N}(S_{i-1}^V) \\ 0 & \text{otherwise,} \end{cases}$$

Vertex selection probability

$$\mathcal{N}(S_{i-1}^V) = \bigcup_{i'=1}^{i-1} \mathcal{N}(v_{t_{i'}}) - \{v_{t_1}, v_{t_2}, \ldots v_{t_{i-1}}\}$$

Methodologies

Graph Dictionary:

Dictionary will store the most commonly occurring subgraphs

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

1. If the universe U (all subgraphs) is small enough to enumerate, we can assume an uniform distribution over U

   $$\mathrm{L}(D) = |D| \log |\mathfrak{U}|.$$

2. If the universe U is too large too enumerate, subgraphs can be stored one-by-one based on the null-model encoding (assign equal probability to all labelled graphs with n vertices and m edges)

$$\mathrm{L}(D) = \sum_{a_i \in D} \mathrm{L}_{\text{null}}(a_i). \qquad L_{\text{null}}(G) = \log(n_{\max} + 1) + \log\left(\binom{n}{2} + 1\right) + \log\binom{\binom{n}{2}}{m},$$

Number of vertices

Number of edges

Number of ways to choose m edges

n: number of vertices
m: number of edges
n_{max}: upper bound of n

Methodologies

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

b_{dict}: $|H_{dict}|$
b_{null}: $|H_{null}|$
q: probability distribution

Graph Encoding:

dual encoding of subgraphs: H_{dict} and H_{null}

- Number of subgraphs: $q_\phi(b_{\text{dict}}, b_{\text{null}}) = \textbf{Binomial}(b_{\text{dict}}|b;\phi)q_\phi(b) = \binom{b}{b_{\text{dict}}}(1-\delta_\phi)^{b_{\text{dict}}}\delta_\phi^{b-b_{\text{dict}}}q_\phi(b),$
- Dictionary subgraphs: $q_\phi(\mathcal{H}_{\text{dict}}|b_{\text{dict}}, D) = \textbf{Multinomial}(b_1, b_2, \ldots, b_{|D|} \mid b_{\text{dict}};\phi) = b_{\text{dict}}! \prod_{a\in D}\frac{q_\phi(a)^{b_a}}{b_a!},$
- Non-Dictionary subgraphs: $q_\phi(\mathcal{H}_{\text{null}}|b_{\text{null}}, D) = \prod_{H_i\in\mathcal{H}_{\text{null}}} q_{\text{null}}(H_i).$
- Cut: $\mathrm{L}(C|\mathcal{H}) = \mathrm{L}(C, m_c, \boldsymbol{m}_c|\mathcal{H}) = \mathrm{L}(m_c|\mathcal{H}) + \mathrm{L}(\boldsymbol{m}_c|m_c, \mathcal{H}) + \mathrm{L}(C|\boldsymbol{m}_c, m_c, \mathcal{H})$

$$= \log\left(1+\sum_{j>i}^{b} k_i k_j\right) + \log\binom{b(b-1)/2 + m_c - 1}{m_c} + \sum_{j>i}^{b}\log\binom{k_i k_j}{m_{ij}}$$

- **(H,C) Length**: $\mathrm{L}_\phi(\mathcal{H}, C|D) = \mathrm{L}_\phi(b_{\text{dict}}, b_{\text{null}}) + \mathrm{L}_\phi(\mathcal{H}_{\text{dict}}|b_{\text{dict}}, D) + \mathrm{L}_{\text{null}}(\mathcal{H}_{\text{null}}|b_{\text{null}}, D) + \mathrm{L}_{\text{null}}(C|\mathcal{H}),$

Methodologies

Graph Encoding:

k_i: vertex count of subgraph H_i
b: number of subgraphs

Cut: $$\mathrm{L}(C|\mathcal{H}) = \mathrm{L}(C, m_c, \boldsymbol{m}_c|\mathcal{H}) = \mathrm{L}(m_c|\mathcal{H}) + \mathrm{L}(\boldsymbol{m}_c|m_c, \mathcal{H}) + \mathrm{L}(C|\boldsymbol{m}_c, m_c, \mathcal{H})$$

$$= \log\left(1 + \sum_{j>i}^{b} k_i k_j\right) + \log\binom{b(b-1)/2 + m_c - 1}{m_c} + \sum_{j>i}^{b} \log\binom{k_i k_j}{m_{ij}}$$

$m_c = \sum_{i<j}^{b} m_{ij}$ (m_c: total cut, m_{ij}: number of edges between each subgraph pair i, j)

$\boldsymbol{m}_c = \{m_{1,1}, m_{1,2}, \ldots, m_{b-1,b}\}$

C will be encoded hierarchically (uniform encoding of all):

1. Encode total cut count m_c
2. Encode pairwise cut count $\boldsymbol{m}_c$
3. Encode the arrangement of the edges

Methodologies

Graph Encoding:

Remarks about graph isomorphism (GI):

- given a fixed decomposition, the parameterization is invariant to isomorphism
  - 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

Optimization and learning algorithms

Dictionary:

- Define x_i indicating whether D contains subgraph a_i: $x_i = \begin{cases} 1 & \text{if } a_i \in D \\ 0 & \text{otherwise.} \end{cases}$
- Continuous relaxation:

$$\hat{x}_i = \sigma(\psi_i), \forall i \in \{0, 1, \ldots, |\mathfrak{U}|\}$$

$$\hat{x}_i \in [0, 1] \text{ a fractional alternative to } x_i$$

→ Then we can optimize using the surrogate gradient w.r.t $\hat{x}_i$

Parametric graph partitioning algorithm:

- Constrain this space via a differentiable parameterization that allows us to perform gradient-based optimization (i.e. GNNs, DeepSets, MLPs, Softmax)

Optimization and learning algorithms

Minimize Total Description Length:

Minimum Description Length (MDL) objective:

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

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

$$\mathrm{L}(D) = |D| \log |\mathfrak{U}|. \quad \text{or} \quad \mathrm{L}(D) = \sum_{a_i \in D} \mathrm{L}_{\text{null}}(a_i).$$

|U| small

|U| large

Optimization and learning algorithms

Final MDL objective:

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

Taking the expectation over the GNN output $(\mathcal{H}, C) \sim p_\theta^{\mathrm{GNN}}(\mathcal{H}, C|G))$, we calculate the gradients as:

$$\nabla_\phi L(\mathcal{G}) = \sum_{G \in \mathcal{G}} \mathbb{E}[\nabla_\phi \mathrm{L}_{\phi,\mathbf{x}}(\mathcal{H}, C)|D)]$$

$$\nabla_{\hat{\mathbf{x}}} L(\mathcal{G}) = \nabla_{\hat{\mathbf{x}}} \mathrm{L}_{\hat{\mathbf{x}}}(D) + \sum_{G \in \mathcal{G}} \mathbb{E}[\nabla_{\hat{\mathbf{x}}} \mathrm{L}_{\phi,\hat{\mathbf{x}}}(\mathcal{H}, C|D)]$$

$$\nabla_\theta L(\mathcal{G}) = \sum_{G \in \mathcal{G}} \mathbb{E}[\mathrm{L}_{\phi,\mathbf{x}}(\mathcal{H}, C|D) \nabla_\theta \ln p_\theta^{\mathrm{GNN}}(\mathcal{H}, C|G)]$$

Theoretical analysis

Compare Partition and Code(PnC) with 2 baselines

Baseline1:Partition

a graph is decomposed into subgraphs and cuts, but the **distribution of subgraphs is not modelled**.

$$L_{\text{part}}(G) = L_{\text{null}}(H) + L_{\text{null}}(C|H)$$

G: Graph H: Subgraph C: Cut

Baseline2:Erdős–Rényi

This method treats all images as unlabeled Erdős–Rényi graphs (ER graphs), and all unlabeled Erdős–Rényi graphs with the same number of vertices n and edges m **will be encoded to the same code**.

$$L_{\text{ER-S}}(G) = \log |G_{n,m}| + \log(n^2 + 1)$$

G: Graph

$G_{n,m}$: The set of all graphs with n vertices and m edges

Theoretical analysis

Consider a distribution p over graphs with n vertices and a partitioning algorithm that decomposes a graph into b blocks of k vertices. Then it holds that:

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

under the following conditions:

(1a) $\log\left(\frac{k^2+1}{k^2}\right) + \overline{H_{m_{ij}}} < \overline{H_m}$ When (1a) hold,then "Partition" better than "ER"

$\overline{H_{m_{ij}}} = \mathbb{E}_{G\sim p}[H(\frac{m_{ij}}{k^2})]$ and $\overline{H_m} = \mathbb{E}_{G\sim p}[H(\frac{m}{n^2})]$ is the expected binary entropy of the cut size m_{ij} between two subgraphs i, j and that of the total number of edges m, respectively.

(1b) $|D| < (k^2+1)2^{k^2\overline{H_{mi}}}$ When (1b) hold,then "PnC" better than "Partition"

$|D|$ is the size of the dictionary and $\overline{H_{m_i}} = \mathbb{E}_{G\sim p}\,[H(\frac{m_i}{k^2})]$ is the expected binary entropy of the number of edges m_i in the subgraph i.

Theoretical analysis

The compression gain:

(1) $$\mathbb{E}_{G\sim p}[L_{\text{part}}(G)] \approx \mathbb{E}_{G\sim p}[L_{\text{ER-S}}(G)] - \frac{n^2}{2}\left(\overline{H_m} - \frac{\log(k^2+1)}{k^2} - \overline{H_{m_{ij}}}\right)$$

(2) $$\mathbb{E}_{G\sim p}[L_{\text{PnC}}(G)] \approx \mathbb{E}_{G\sim p}[L_{\text{part}}(G)] - nk(1-\delta)\left(\overline{H_{mi}} - \frac{H(D) - \log(k^2+1)}{k^2}\right)$$

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

How to get the compression gain:

(1) Large k, low $\overline{H_{m_{ij}}}$ ⟶ Subgraph as large as possible, cut edge as less as possible

(2) Large k, low $H(D)$ ⟶ Subgraph as large as possible, dictionary need to be reasonably chosen

Theoretical analysis （Prove Partition's compression gain from ER）

Consider a distribution p over graphs with n vertices and a partitioning algorithm that decomposes a graph into b blocks of k vertices:

$$L_{\text{part}}(G) = L_{\text{null}}(H) + L_{\text{null}}(C|H)$$

G: Graph H: Subgraph C: Cut

$$\mathbb{E}_{G\sim p}[\mathrm{L}_{\text{part}}(G)] \approx \mathbb{E}_{G\sim p}\left[\sum_{i=1}^{b}\left(\log(k^2+1)+k^2\mathrm{H}(\frac{m_i}{k^2})\right)+\sum_{i\neq j}^{b}\left(\log(k^2+1)+k^2\mathrm{H}(\frac{m_{ij}}{k^2})\right)\right]$$

$\log(k^2+1)$:encode all possible edges

$k^2\mathrm{H}(\frac{m_i}{k^2})$:encode actual edges

$$\log\binom{k^2}{m_i} \approx m_i \log\left(\frac{k^2}{m_i}\right) + (k^2-m_i)\log\left(\frac{k^2}{k^2-m_i}\right)$$

$$= \frac{n}{k}\left(\log(k^2+1)+k^2\bar{\mathrm{H}}_{m_i}\right)+(\frac{n^2}{k^2}-\frac{n}{k})\left(\log(k^2+1)+k^2\bar{\mathrm{H}}_{m_{ij}}\right)$$

$$= n^2\left(\frac{\log(k^2+1)}{k^2}+\bar{\mathrm{H}}_{m_{ij}}\right)+nk\left(\bar{\mathrm{H}}_{m_i}-\bar{\mathrm{H}}_{m_{ij}}\right)$$

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

$$+nk\left(\bar{\mathrm{H}}_{m_i}-\bar{\mathrm{H}}_{m_{ij}}+\log n\right)-\log(n^2+1)$$

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

$$\overline{H_{m_i}} = \mathbb{E}_{G\sim p}[H(\frac{m_i}{n^2})]$$

$$\overline{H_{m_{ij}}} = \mathbb{E}_{G\sim p}[H(\frac{m_{ij}}{k^2})]$$

$$L_{\text{ER-S}}(G) = \log|G_{n,m}| + \log(n^2+1)$$

$$\approx n^2\mathrm{H}_m + \log(n^2+1) - n\log n$$

Eliminate duplicate vertex permutations

Theoretical analysis（Prove PnC's compression gain from Partition）

Consider a distribution p over graphs with n vertices and a partitioning algorithm that decomposes a graph into b blocks of k vertices:

For the number of dictionary subgraph:

b_{dict}:the number of subgraphs in the dictionary

$1-\delta$:the probablity that a subgraph belongs in the dictionary

$$\mathbb{E}_{G\sim p}[\mathrm{L}(b_{\mathrm{dict}})] = \frac{1}{2}\log\Big(2\pi e b\delta(1-\delta)\Big) + O\Big(\frac{1}{b}\Big)$$

$$= \frac{1}{2}\log\Big(2\pi e\delta(1-\delta)\Big) + \frac{1}{2}\log\Big(\frac{n}{k}\Big) + O\Big(\frac{k}{n}\Big) = O\Big(\log\frac{n}{k}\Big)$$

$b_{dict}\sim Binomial(b, 1-\delta)$

$$H(b_{\mathrm{dict}}) \approx \frac{1}{2}\log(2\pi e b\delta(1-\delta))$$

For dictionary subgraph:

The expected length of the subgraphs that belong in the dictionary amounts to the entropy of the **multinomial** distribution

$$\mathbb{E}_{G\sim p}[\mathrm{L}(\mathcal{H}_{\mathrm{dict}}|b_{\mathrm{dict}}, D)] = \mathbb{E}_{G\sim p}\left[-\log\frac{b_{\mathrm{dict}}!}{\prod_{a\in D} b_a!} - \sum_{a\in D} b_a \log q(a)\right]$$

$$\leq \mathbb{E}_{G\sim p}[-\sum_{a\in D} b_a \log q(a)]$$

$$= -\sum_{a\in D} \mathbb{E}_{G\sim p}[b_a]\log q(a)$$

$$= -b(1-\delta)\sum_{a\in D} q(a)\log q(a) = \frac{n}{k}(1-\delta)\mathbb{H}(D) \leq \frac{n}{k}(1-\delta)\log|D|,$$

b_a:the frequence of subgraph a

$q(a)$:the probability of subgraph a exists in the dictionary

$$\mathbb{E}_{G\sim p}[b_a] = b(1-\delta)q(a)$$

Theoretical analysis （Prove PnC’s compression gain from Partition）

For Graph:

$$\mathbb{E}_{G\sim p}[\mathrm{L}_{\mathrm{PnC}}(G|D)] \lessapprox \mathbb{E}_{G\sim p}[\mathrm{L}_{\mathrm{part}}(G)] + n(1-\delta)\left(\frac{\mathbb{H}(D) - \log(k^2+1)}{k} - k\bar{\mathrm{H}}_{m_i}\right) + O(\log\frac{n}{k})$$

:for subgraphs in the dictionary. Do not need to encode their edges

Proof Finished!

Empirical Results

Datasets: small molecules, proteins and social networks

Baseline: NULL models: No parameters

(1) Uniform model: all edges are assumed to be sampled independently with probability equal to 0.5.

(2) Edge list model: Lists all the edges in the graph in order, storing the graph structure as edges.

(3) Erdős–Rényi

Partitioning-based: grouping vertices in tightly-connected clusters

(1) SBM fitting: assuming there is a hidden community structure in the graph and the SBM is fitted by Bayes method.

(2) Louvain clustering: Hierarchical clustering a graph.

(3) Label Propagation: using vertex label propagation to cluster nodes in the graph.

Empirical Results

Baseline: Likelihood-based neural compressors: With numerous parameters

(1) GraphRNN：a graph generation model based on autoregression, is suitable for generating chain structures.

(2) GRAN: Graph Recurrent Attention Network, can generate complex graph structures.

Metrics: Average bits per edge (bpe)

Empirical Results

data: cost of compressing the data

total: total cost (including params)

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

PnC can obviously decrease the cost of compression

GraphRNN and GRAN is good at compressing data, but use too many parameters

PnC + Neural can improve the cost of data compression and decrease the number of params

Empirical Results

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

SBM has good performance, because it's good at finding community structures

PnC + Neural have best performance on Proteins, because neural networks can capture complex graph structures

Null models have worst performance on all datasets

References

Abbe, E. (2016). Graph compression: The effect of clusters. *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 1–8. https://doi.org/10.1109/ALLERTON.2016.7852203

Bouritsas, G., Loukas, A., Karalias, N., & Bronstein, M. (2021). Partition and Code: Learning how to compress graphs. *Advances in Neural Information Processing Systems*, *34*, 18603–18619. https://proceedings.neurips.cc/paper/2021/hash/9a4d6e8685bd057e4f68930bd7c8ecc0-Abstract.html

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pp. 3391–3401, 2017.

Thank you!