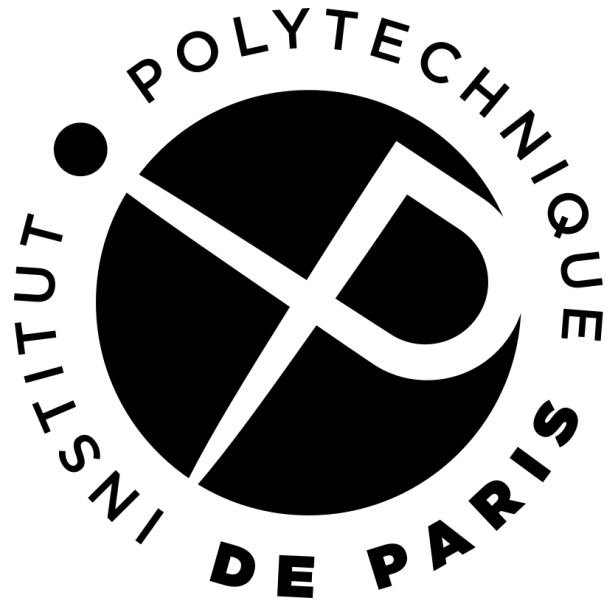
Improving Graph Generation by Restricting Graph Bandwidth

GERTNER Victor, ZERBIB Yohann

August 13, 2024



Contents

1 Introduction

Generative Graph Modeling is a subfield of machine learning that aims to generate graph or to predict missing part of graphs, which is particularly useful in domains such as molecular chemistry or when representing social networks. However, a typical problem one can encounter when trying to implement deep graph generative modeling is the fact that the output space is very large and it compromises the performance of the model. Moreover, it is often not relevant as concrete applications do not need such broad output spaces. Therefore, the paper that we chose to study, "Improving graph generation by restricting graph bandwidth" by Diamant et al. [1] focuses on reducing the length of this bandwidth and see if it fits better the expectations for many real-world problems. In this paper analysis, we would like to explain the proposed approach, to assess it and to try to replicate it on another dataset.

2 Study of the proposed approach

2.1 Task Description

The framework this paper takes place is the field of structured data, one would try to capture the underlying distribution of a structured object. More precisely, the task consists in implementing a bandwidth restriction strategy. Let's explain it little by little. The framework of this problem was defined by Unger [2].

Let $G = (V, E)$ be a graph with $|V| = n$ vertices, and let A be its adjacency matrix. Let π be a permutation of the vertices of G, and let A_π be the adjacency matrix of G with its rows and columns permuted according to π. The bandwidth of G with respect to π is defined as:

$$\text{bandwidth}(G, \pi) = \max\{|i - j| : A_\pi(i, j) \neq 0\} \tag{1}$$

In other words, the bandwidth of G with respect to π is the maximum difference between the row and column indices of any non-zero entry in the permuted adjacency matrix A_π. The bandwidth of G is then defined as the minimum bandwidth over all possible permutations of its vertices:

$$\text{bandwidth}(G) = \min\{\text{bandwidth}(G, \pi) : \pi \text{ is a permutation of } V\} \tag{2}$$

An equivalent definition given in the article is the following, using the notion of ordering.

An ordering of G is a bijection $\pi : V \to [1, N]$. The distance between two nodes v_i and v_j in the ordering π is defined as $\text{dist\_}\pi(v_i, v_j) = |\pi(v_i) - \pi(v_j)|$. The bandwidth of the ordering π is defined as the maximum distance between any two adjacent nodes, i.e. $\phi(\pi) = \max_{(v_i,v_j)\in E} \text{dist\_}\pi(v_i, v_j)$. The bandwidth of the graph G is defined as the minimum bandwidth over all possible orderings, i.e.

$$\phi(G) = \min_{\pi: V \to [1,N]} \phi(\pi). \tag{3}$$

Thus, having a an ordering that minimize this problem can be used to transform the representation of the adjacency matrix from N*N to N*ϕ.

2.2 Solution to the task : Cuthill-McKee Algorithm

The problem described by (3) is known to be NP-Hard in the general case, and that exact polynomial solution only exist for a really restricted type of graphs. This is why, this paper uses an heuristic approach (i.e approaches that may not guarantee the optimal solution, but that often produce solutions that are acceptable in practical applications.)

Cuthill-McKee Algorithm :

In real world graphs, there is an high number of nodes N, this leads to a adjacency matrix of size N^2, but in the same time, there is only a small number of edges that appear from those N^2 possibilities. This leads to having sparse adjacency matrix where only a few number of values are not at 0. Knowing this we can leverage methods from sparse matrix optimisation.

The Cuthill McKee algorithm is a variant of the standard breadth-first search algorithm used in graph algorithms.

Algorithm 1 Cuthill-McKee Algorithm

1: Start with an adjacency list representation of the graph.
2: Initialize an empty queue.
3: Choose a starting vertex (usually the one with the lowest degree).
4: Enqueue the starting vertex.
5: **while** the queue is not empty **do**
6: Dequeue a vertex from the queue.
7: Visit its neighbors in increasing order of their degrees.
8: Enqueue each unvisited neighbor.
9: **end while**
10: Once all vertices have been visited, assign a new index to each vertex based on the order they were visited.
11: Reorder the adjacency matrix according to the new indices.
12: The reordered matrix has reduced bandwidth.

We easily see that this algorithm has a linear time complexity O($|\epsilon|$).

We can see from this exemple that the C-M algorithm consistently reduces $\psi(\pi)$ compared to the orderings routinely used in graph generation (BFS, DFS, etc.).

Next in this project, we define the bandwidth of the adjacency matrix derived with the C-M algorithm as $\hat{\psi}$.

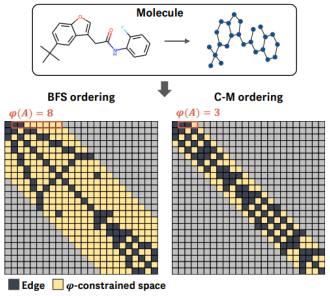


Figure 1: Bandwidth of two adjacency matrices of the same molecular graph. The left adjacency matrix is given by a BFS ordering, the right adjacency matrix is given by a Cuthill-McKee ordering.

2.3 Bandwidth-Restricted Graph Generation

Now that we tackled the task to restrict the Bandwidth of graphs, it's now time to understand how this can be implemented in any generative methods and what are its benefits.

First we will look at the general case implementation and then we will look at real end to end solution that can be upgraded with the Bandwidth-Restricted representation.

2.3.1 General case :

In the general case, a generative model tries to lean the distribution of the data $p(G)$, the ouput space in general case is of size N*N, but with Bandwidth-Restricted representation we can constraint the output space to be of size $N * \varphi$ without losing expressiveness. This means that the graph that are output still exhibit similar statistical properties, structures, and features as the training data.

General Training : During training, we consider the adjacency matrix $A^{\pi_{C-M}}$, where π_{C-M} denotes the ordering done by the Cuthill-McKee algorithm. One thing to note is that, since the complexity of the Cuthill-McKee algorithm is linear, we do not introduce more complexity compared to other adjacency matrix ordering methods (BFS, DFS...). But this is not the only adjustment we make to the normal training procedure : It was shown in [3] that constraining training example to a certain ordering doesn't force the model to output data having the same ordering. Moreover, it doesn't improve neither time or space complexity, since we here are still considering the full matrix. This is why a new step is mandatory for the representation to be useful : we transform the $A^{\pi_{C-M}} \in N * N$ to a $A^{\pi_{C-M}}_{opt} \in N * \hat{\varphi}$. To do so, we drop the zeros outside the bandwidth.

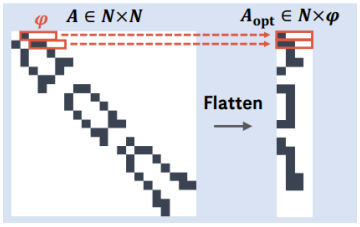


Figure 2: Transformation of $A^{\pi_{C-M}}$ into $A^{\pi_{C-M}}_{opt}$

Hence, with this constraint, only only edges belonging to the reduced matrix are considered as candidates, thus constraining the output space and reducing the time complexity, without losing expressiveness.

Every model we will come across will try to predict $p[(v_i, v_j) \in \epsilon]$ for each node such that $\pi(v_j) < \pi(v_i)$

2.3.2 Autoregressive Graph Generation with Bandwidth-Restricted Graph

In an autoregressive setup, the model generate the adjacency matrix row-by-row or block-by-block with a recursive approach. The focus in the paper is made on GraphRNN.

Training : We train the model to predict $p[(v_i, v_j) \in \epsilon]$ by using an RNN and an output function f_θ so that $p[(v_i, v_j) \in \epsilon] = f_\theta(RNN_\phi(A^\pi_{1:i-1}))_j$, where θ and ϕ are parameters learned to maximize the likelihood of the data.

We observe that a d-unit f_θ has the capability to generate graphs with a maximum bandwidth of d. While one approach could be setting d to the maximum number of nodes, N, of any graph we aim to generate, ensuring maximum expressiveness, we opt instead to set d equal to the maximum bandwidth of any A_π we intend to generate. This choice significantly enhances efficiency and diminishes training signal sparsity, particularly for low bandwidth graphs.

Determining the order π for each graph G involves employing the C-M algorithm. We then set d to $\hat{\phi}$data, representing the maximum bandwidth across all $A\pi$ in the training data. This approach allows for generating significantly fewer edges, approximately $O(N/\hat{\varphi}_{\text{data}})$ times less compared to generating N rows of length $d = N$.

GraphRNN was trained using teacher forcing to autoregressively predict the next row of the re-parameterized adjacency matrices $A_{\text{opt}} \in \mathbb{N} \times \hat{\phi}_{\text{data}}$, by having the data preprocess as such :

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & A_{\text{opt},0,0} & \cdots & A_{\text{opt},0,\hat{\varphi}_{\text{data}}} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & A_{\text{opt},N,0} & \cdots & A_{\text{opt},N,\hat{\varphi}_{\text{data}}} \\ 1 & 0 & \cdots & 0 \end{bmatrix}$$

Thus, the training is made using stochastic gradient descent with a maximuml ikelihood loss over S^{π}i.e., we optimize the parameters of the neural networks to optimize $\prod p_{model}(S^{\pi})$ over all observed graph sequence.

Inference Step : The rows of the constructed data matrix were sampled based on the logits, denoted as l, which were output from the model's final layer and adjusted by a temperature parameter. This process resulted in edge probabilities for row i, denoted as $p[(v_i, v_j) \in \epsilon]$, following a Bernoulli distribution with parameter $\frac{1}{\tau} l_{i,j+1}$.

2.3.3 One-shot Graph Generation with Bandwidth-Restricted Graph

One-shot graph generative models generate the entire graph topology at once. This is often achieved by assigning edge probabilities to every pair of nodes, effectively creating a complete graph where the generative model places a probability on each edge. Additionally, we refer to this graph as the edge-probability graph. The main insight is that the full edge probability graph can be substituted with a bandwidth-constrained edge-probability graph. We will only tackle the part Node-embedding-based model from the article.

Node-embedding-based models generate node embeddings from a latent distribution and then derive the edge-probability graph by considering pairwise relationships. The proposed model to look at is Graphite [4]

Graphite is taking advantage of the VAE architecture :

$$\mu, \sigma = \mathrm{GNN}_{\phi}(A, X) \tag{5}$$

Adjacency matrix $A \in \mathbb{R}^{N \times N}$ and node features $X \in \mathbb{R}^{N \times k}$.

The decoder network outputs edge probabilities:

$$p[(v_i, v_j) \in E] = \mathrm{GNN}_{\theta}(A, X, \hat{Z})_{i,j} \tag{6}$$

$\hat{A}$ is fully-connected and Z are samples from $\mathcal{N}(\mu, \sigma)$.

In the bandwidth-restricted version of Graphite, the bandwidth of $\hat{A}$ is constraint . For a graph G, we get the bandwidth $\varphi(A_{\pi})$ for π found using the C-M algorithm and build a new edge set:

$$\hat{E} = \{(i, j) \mid 1 \leq |i - j| \leq \varphi(A_{\pi})\},$$

where $1 \leq i, j \leq N$.

With the introduction of the new edge set, the computational complexity of the decoder's message passing steps decreases from $O(N^2)$ to $O(N \cdot \hat{\varphi})$.

3 Metrics

There are two things to take care on a generation task :

- **closeness of the sample distribution to the true distribution of graphs**
- **reconstruction quality**

3.1 Closeness of the sample distribution to the true distribution of graphs

In order to measure the closeness of the sample distribution to the true distribution of graphs two metrics are used :

Maximum Mean Discrepancy It measures the discrepancy between two distributions : the distribution of sampled graphs and the true distribution of graphs.

Suppose that a unit ball in a reproducing kernel Hilbert space (RKHS) H is used as its function class F, and k is the associated kernel, the squared MMD between two sets of samples from distributions p and q can be derived as :

$$MMD^2(p||q) = \mathbb{E}_{x,y\sim p}[k(x,y)] + \mathbb{E}_{x,y\sim q}[k(x,y)] - 2\mathbb{E}_{x\sim p,y\sim q}[k(x,y)]. \tag{7}$$

Proper distance metrics over graphs are in general computationally intractable. Thus, we compute MMD using a set of graph statistics $M = \{M_1, \ldots, M_k\}$, where each $M_i(G)$ is a univariate distribution over $\mathbb{R}$, such as the degree distribution or clustering coefficient distribution. We then use the first Wasserstein distance as an efficient distance metric between two distributions p and q:

$$W(p,q) = \inf_{\gamma\in\Pi(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[\|x-y\|], \tag{8}$$

where $\Pi(p,q)$ is the set of all distributions whose marginals are p and q respectively, and γ is a valid transport plan.

The kernel function defined by $k_W(p,q) = \exp\left(\frac{W(p,q)}{2\sigma^2}\right)$ is used.

The graph statistics compared are degree, clustering coefficient, node orbit counts and spectrum.

F1-PR Score Recent recommendations for evaluating graph generative models employ a precision-recall metric that considers both sample quality and variety. The harmonic mean of precision and recall is reported as the F1-PR score to provide a balanced assessment of the model's performance.

3.2 Reconstruction quality

The quality of graph reconstruction is evaluated by comparing the reconstructed graph with the original test graph using the Area Under the Precision–Recall Curve (AUPRC). For GraphRNN, the comparison is done between the reconstructed row and the original row, while for Graphite, the comparison is between the reconstructed graph and the original graph. AUPRC is chosen because it is independent of true negatives and is suitable for class imbalance. Additionally, the estimated log-likelihood for test graphs can be reported as it is useful in demonstrating the advantages of BwR's reduced output space when combined with other metrics.

4 Paper assessment

One could shed light on different aspects of this paper to assess it. First of all, it is true that the idea of reducing the size of bandwidth is cogent, and well-founded on the real-world issue. Here, what is aimed is not an innovation but rather a new approach of an existing graph generation problem. The originality here lies in the idea of restricting graph-bandwidth. One should highlight that techniques of ordering already existed but the aggregation for this case was made by the authors. The paper is clear as well and the experiments are well-argued, described. The metrics that are used make sense, and their are several of them that divide what is expected in different parts (reconstruction / generation) to be sure to avoid missing any important objective of the model. However, we were not sure about the complexity at some point. In the paper, some affirmations are made about reducing the complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N\phi)$, but it is unsure it takes into account the complexity of the model considered. There might be changes in complexity during training and inference for the models proposed in the article (e.g. VAE complexity without bandwidth reduction and then with bandwidth reduction during training and also during inference). It seems also arbitrary why the CM algorithm is privileged over the reverse-CM algorithm when there are several situations where the reverse version seem to perform better generally. A mention could have been interesting. Finally, the reproducibility is a complicated point. It is true that the whole work is on github with a conda environment setup. However, there seem to be many part in the code that are barely used. Some metrics require a mixed implementations with C++. Models can be long to train and require GPU. Therefore, it cannot be easily done locally (the conda environment seems to require a special system distribution as well). The results are mixed, because the github remains open-source and there are many things that can be taught from it with time.

5 Experiments

In this section, experiments that we tried to replicate will be introduced. Firstly, it should be highlighted that even if a github was provided in the article, cloning it to test it requires a GPU and the proposed environment did not seem to work properly. It should also be highlighted that "orca" is built for the evaluation of the dataset, and requires C++, which seemed hard to transfer to notebooks / colab. Those two considerations heavily limited the possible options. Therefore, it was decided to focus on the easier model to train, "graphRNN". When one tried to synthetize the essence of the approach for graphRNN, it all came down to the following step.

1. Use the cuthill-mckee algorithm to get a node order for each graph.
2. Convert graphs into adjacency matrices using the node orders.
3. Find the maximum estimated bandwidth from those adjacency matrices, then truncate the rows of the adjacency matrix to only be max estimated bandwidth in length.
4. Train an RNN to predict the next truncated row of the adjacency matrix given the previous rows.

However, it is difficult to evaluate the training of this model because we lack the implementation of the metrics that are not easily available as explained. The dataset on which we worked is the QM9 dataset. The QM9 dataset is a collection of small organic molecules with associated quantum mechanical properties. It's commonly used as a benchmark dataset in quantum chemistry and machine learning research. It was split for training and test purpose.
Another issue faced is the computation time of our code. It was quite long (even when putting the objects on cuda) and therefore it was difficult to try to optimize it, it should be considered as a demo of the idea. About the implementation, the aim was too be close from the article, adapted on a colab notebook. It was mandatory to pad matrices to have a working implementation, therefore the preprocessing differs from the article.

At the end, the model is successfully (even if it takes much time) train but we lacked time and resources to evaluate it properly.

References

[1] Nathaniel Lee Diamant, Alex M Tseng, Kangway V Chuang, Tommaso Biancalani, and Gabriele Scalia. Improving graph generation by restricting graph bandwidth. In *International Conference on Machine Learning*, pages 7939–7959. PMLR, 2023.

[2] Walter Unger. The complexity of the approximation of the bandwidth problem. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 82–91. IEEE, 1998.

[3] Han X. Hu J. Ruiz F. Chen, X. and L Liu. Order matters: probabilistic modeling of node sequence for graph generation. In *Proceedings of the 38th International Conference on Machine Learning*, page 1630–1639, 2021.

[4] Zweig A. Grover, A. and S Ermon. Graphite: iterative generative modeling of graphs. In *Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research*, page 2434–2444, 2019.