Synthesizing and Scaling WAN Topologies using Permutation-invariant Graph Generative Models

Max Helm, Georg Carle
Department of Computer Engineering
Technical University of Munich
Munich, Germany
{helm, carle}@net.in.tum.de

***Abstract*—Real-world Wide Area Network (WAN) topologies are scarce. The shift towards machine learning in network management and optimization brings a need for large datasets, including real-world topologies. WAN topologies can be generated using graph generative models. Graph generative models can be divided into parameterized and data-driven approaches. Data-driven approaches can be further divided into permutation-invariant and permutation-variant. In this paper, we improve on existing work, which utilized adjacency-matrix-based, permutation-variant Generative Adversarial Networks to synthesize WAN topologies. We achieve this by using existing, data-driven approaches that are permutation-invariant w.r.t. their input. Our results show a decrease in the mean Kolmogorov-Smirnov distance over various graph theoretical metrics of 80%. Furthermore, we employ graph upscaling models to increase WAN topology sizes while preserving their properties up to a scaling factor of 256. We publish all datasets and hope they can be of help in training machine learning models, such as communication network performance prediction models or digital twins, enabling better automated network management.**

***Index Terms*—graph; wide area networks; generative machine learning**

I. Introduction

The need for large and diverse datasets of communication networks is increasing due to new applications of machine-learning to network control and optimization. An integral part of a dataset are the network topologies. Especially real-world Wide Area Network (WAN) topologies are a limited resource [1]. Related work tries to alleviate this lack of data by employing Generative Adverserial Networks (GANs) to generate realistic WAN topologies [2].

The goal of this work is to give an overview over different existing methods to achieve the same goal, benchmark them against each other, and give recommendations wich method produces the best results. Additionally, we provide a dataset produced by the best-performing method.

Another drawback of real-world WAN topologies is that they are limited in size [1]. Often, we want to evaluate machine-learning models on their ability to generalize to larger input spaces, including the topology size. To this end we compare three existing graph up-scaling approaches. Furthermore, we provide a dataset of larger topologie based on real-world WANs that retain their topological structure for scaling factors of up to 256.

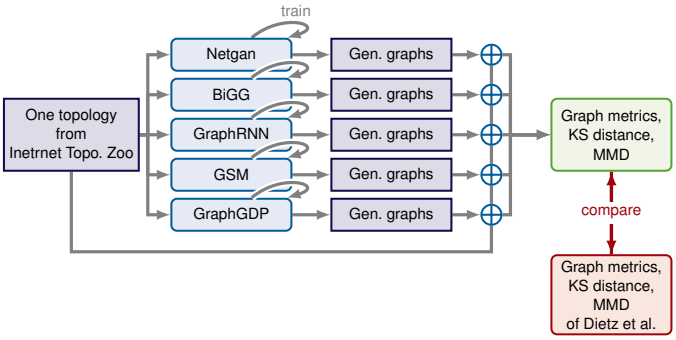


Fig. 1: Workflow of comparing different graph generation approaches.

The remainder of the paper is structured as follows. Section II provides an overview of data-dirven methods that are used to generate and scale networks. Section III gives an overview of related work in WAN generation. Section IV details our methodology for comparing different approaches. Section V provides the results of the graph generation and up-scaling benchmarks. Section VI shows some limitations of this work with notes on reproducibility in Section VII and concluding remarks in Section VIII.

II. Background

This section introduces background information about WANs and generative machine learning methods.

A. Internet Topology Zoo

The Internet Topology Zoo [1] is a collection of Wide Area Network (WAN) topologies from real deployments around the world, while concentrating mostly on Europe and North America. It contains over 250 topologies and is regularly used as data source for publications, e.g., as input for machine-learning methods [8], [9].

B. Generative Adversarial Networks

Generative Adversarial Networks (GANs) [10] are a machine-learning approach to estimating the underlying distribution from which a set of samples is drawn. They are based on game theory and pit two models, with different optimization goals, against each other. They are mostly used for image synthesis.

978-3-903176-59-1 ©2023 IFIP

TABLE I: Five selected graph generation methods, each based on different underlying approaches, and their capabilities.

Name	Year	Reference	Approach	Permutation Invariant	Directed	Edge Weights	Node Feat.
Netgan	2018	[3]	Random walks	✓	✗	✗	✗
GraphRNN	2018	[4]	Autoregressive model	BFS-order	✗	✗	✗
GraphScoreMatching	2020	[5]	Graph score matching with GNNs	✓	✗	✗	✗
BiGG	2020	[6]	Autoregressive tree model	DFS-/BFS-order	✓	✗[1]	✓
GraphGDP	2022	[7]	Diffusion model	✓	✗	✗	✗

[1]Possible to indirectly include edge weights as node features

C. Graph Neural Networks

Graph Neural Networks (GNNs) [11] are a machine-learning approach that works directly on graph-structured data. The main advantage is that this approach is permutation invariant w.r.t. the input, meaning the nodes and edges in the graph can be in any order without influencing the result. This means we can avoid costly data augmentation methods, such as generating all adjancency matrices for a graph, which would otherwise be needed in order to cover the whole input space. The number of adjacency matrices of a graph is $n!$ with n being the number of nodes in the graph.

GNNs work on a graph in the form of a set of vertices and edges $G = (V, E)$. They utilize the message-passing paradigm to propagate information through the graph. It consists of message passing, aggregation, and update. During message passing, the hidden state of a node is sent to all neighbors. Then, these hidden states from neighbors are aggregated at each node. Finally, the hidden state of each node is updated using a function of its own hidden state and the aggregated hidden states from its neighbors.

D. Diffusion Models

Diffusion models [12] are a machine learning approach that consists of a Markov chain. The Markov chain is trained to reverse a noise-adding process. This can be used to, for example, generate realistic images from images of gaussian noise.

E. Kronecker Product

The kronecker product is a matrix operation on two matrices of arbitrary sizes. Iterative application of the kronecker product can be used to scale the size of graphs [13].

III. Related Work

Dietz et al. [2] compare a GAN-based approach to classic, non-data-driven approaches on the task of WAN topology synthesis. They implemented a GAN and encoded topologies as pixel images of their adjacency matrices. Adjacency matrices are permuted to approximate permutation invariance. They consider distances between nodes as edge weights encoded as RGB-values of the pixels. Postprocessing is done base on a Bernoulli model with consideration for the symmetry of adjacency matrices. The evaluation is based on the Betweenness Centrality (BC), Closeness Centrality (CC), Degree Centrality (DC), and the Kolmogorov-Smirnov distance between their distributions. They conclude that the GAN approach is flexible and can approximate networks well. In contrast, they note that it struggles with permutations in adjacency matrices and that it requires complex posprocessing. This work indentifies and applies existing methods to alleviate these drawbacks. Namely, we show that permutation invariance can be achieved using related work and postprocessing is not necessary.

Other approaches such as ErdősRényi [14], Barabási-Albert [15], Watts-Strogatz [16], or Scale-free Clustering [17] rely on very few parameters and have shortcomings in accuracy. They are commonly used to generate graph datasets in the communication networks domain. We refer to [2] for a more in-depth analysis.

Leskovec et al. [13] apply a Kronecker graph up-scaling method to autonomous system graphs.

IV. Methodology

In this section, we explain the approach for generating and scaling graphs. Furthermore, we show how we compare the quality of results.

A. Graph Generation

We compare five different machine-learning-based methods for graph generation. Table I details all approaches. The reference data source for all generation approaches is the Internet Topology Zoo.

Netgan [3] is based on the GAN architecture with a generator and a discriminator. They utilize random walks on graphs to generate new graphs. The discriminator needs to distinguish between true graphs and generated graphs solely based on given random walks over the respective graphs. The approach is permutation invariant due to the nature of the random walks.

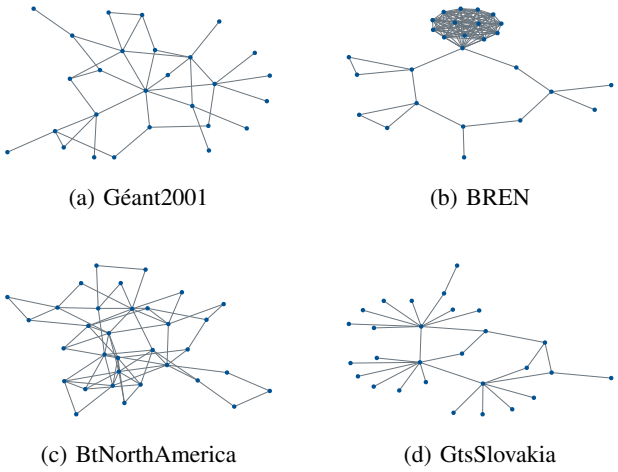
GraphRNN [4] is an autoregressive model. It is not permutation invariant. However, the authors achieve weak permutation invariance by collapsing graphs to their Breadth-first search (BFS) tree representations.

GraphScoreMatching (GSM) [5] relies on graph score matching and GNNs to generate new graphs.

BiGG [6] is another approach relying on autoregression. They achieve weak permutation invariance in a similar way as GraphRNN, by considering the Depth-first- and Breadth-first-search (DFS and BFS) representations of graphs.

GraphGDP [7] is an approach that relies on a diffusion model. They achieve permutation invariance by relying on a permutation equivariant edge prediction which results in a permutation invariant log-likelihood function.

To enable a fair comparison we ensure two things. First, we do not perform any hyperparameter optimization on any of the models. Most of the models have examples defined for



(a) Géant2001 (b) BREN

(c) BtNorthAmerica (d) GtsSlovakia

Fig. 2: Four selected topologies from the Internet Topology Zoo

the `community-small` dataset. We take the hyperparmeters selected by the authors for this dataset, since it is similar in size to the Internet Topology Zoo. Second, we concentrate on undirected graphs without node labels or edge weights since this is the largest subset supported by all models.

The workflow of the comparison approach is shown in Figure 1. Starting with a single topology, we train each of the five models. The early stopping method is either a fixed number of epochs or another criterion, for example, the validation accuracy or the edge overlap. Next, we use each model to generate a set of 100 topologies. We compare these topologies based on graph metrics and their distribution distances (Kolmogov-Smirnov and Maximum Mean Discrepancy) as explained in Section IV-C.

B. Graph Up-scaling

The goal of graph up-scaling is to generate a graph that is larger in size than a given graph while retaining graph theoretical properties. We compare three different graph upscaling methods on the Internet Topology Zoo. They are listed in Table II.

Kronecker [13] is based on the kronecker product that can be calculated over matrices of different sizes.

Gscaler [18] is based on an adaption of DNA shotgun sequencing.

EvoGraph [19] is based on a preferential edge attachment mechanism.

The workflow of comparing these methods on the Internet Topology Zoo is shown in Figure 3. We train each method on a single topology. Next, we generate graphs with each method that are of size $s(G') = s(G) \cdot 2^n \ \forall n \in \{1, 2, 3, 4, 5, 6, 7, 8\}$ with $s(G)$ being the size of the original graph.

The generated graphs of different scales are then compared to the original topology using graph metrics and distribution distances as explained in Section IV-C.

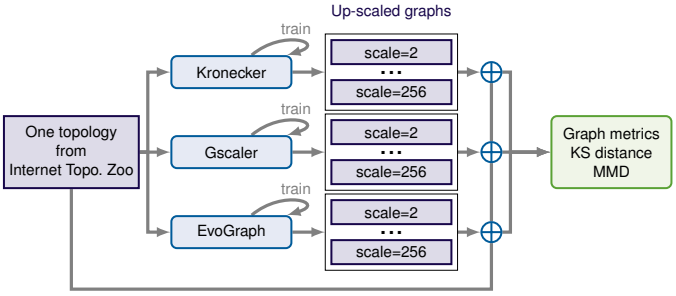


Fig. 3: Comparison methodology for three graph upscaling methods

TABLE II: Three approaches to scale up graphs while preserving graph properties

Name	Year	Reference	Approach
Kronecker	2010	[13]	Kronecker product matrix operation
Gscaler	2016	[18]	DNA shotgun sequencing variation
EvoGraph	2018	[19]	Preferential edge attachment

C. Quality Comparison

We employ two different comparison techniques based on related work.

1) Graph Metrics: The first comparison is based on graph metrics. We directly compare the graph metrics of generated graphs to the baseline graphs. Less difference, while maintaining non-isomporphism, indicates a better generative model. The types of metrics are taken from two areas of related work.

WAN Synthesis We compare all approaches in the same way as [2], using the Betweenness Centrality (BC), Closeness Centrality (CC), and Degree Centrality (DC).

Graph Generation and Upscaling We compare all approaches using graph metrics typically employed in graph generation and graph upscaling papers. These metrics are the clustering coefficient, orbit count, motif, spectra of Laplacian Eigenvalues, triangle count, wegdge count, claw count, assortativity, and gini coefficient.

2) Distance Measures: The second comparison is based on distance measures of the distributions of the graph metrics between the generated and baseline graphs.

WAN Synthesis We compare all approaches using the same method as [2], the Kolmogorov-Smirnov (KS) distance. This distance compares the distribution of BC, CC, and DC values of a generated graph to the distribution of the same values in the baseline graph. This ensures that graph structure is maintained over the entire graph, not only in the mean over all nodes.

Graph Generation and Upscaling We use the Maximum Mean Discrepancy (MMD) over all graph metrics to compare graphs. The MMD is a kernel-based approach to estimate the closeness of two distributions.

V. Evaluation

In this section we evaluate the results of the comparison of graph generation and graph up-scaling methods.

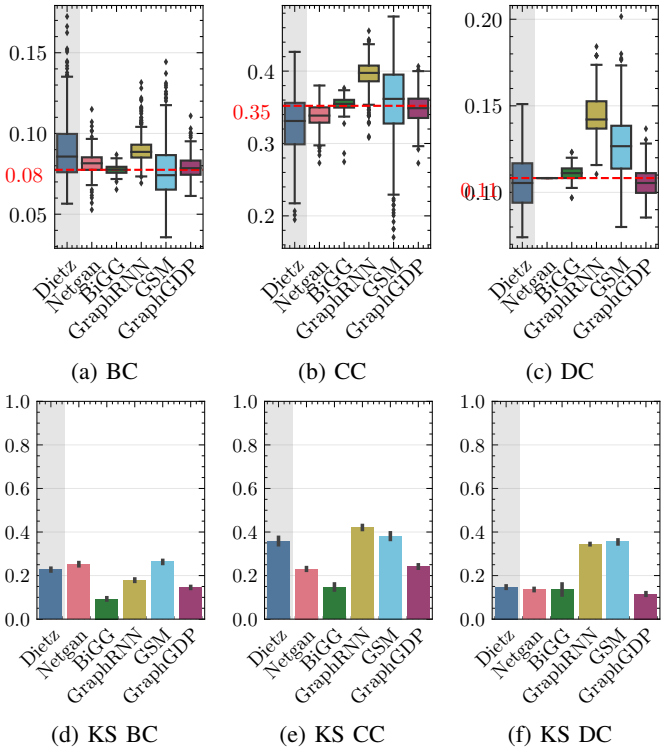


Fig. 4: Synthesis for the Géant2001 topology. Baseline model is highlighted in gray. Target metric (derived from the original Géant2001 topology) is indicated as a red dashed line. Top row shows basic graph metrics (the closer to the red dashed line, the better), bottom row shows the distance in the distribution of graph metrics (lower is better).

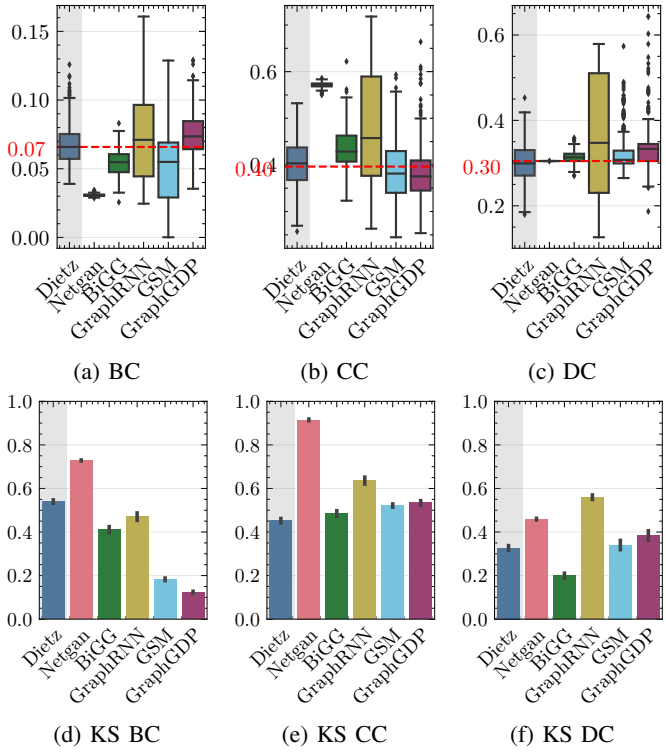


Fig. 5: Synthesis for the BREN topology. Baseline model is highlighted in gray.

A. Graph Generation

To evaluate the quality of generated graphs, we use the metrics defined in Section IV-C.

First, we concentrate on four topologies from the Internet Topology Zoo. We do this to have a direct comparison to Dietz et al. [2] (noted as Dietz for brevity in all comparisons). The four topologies (Géant2001, BREN, BtNorthAmerica, GtsSlovakia) are shown in Figure 2. For each of the four graphs we train each of the five generation methods and generate a set of 100 graphs.

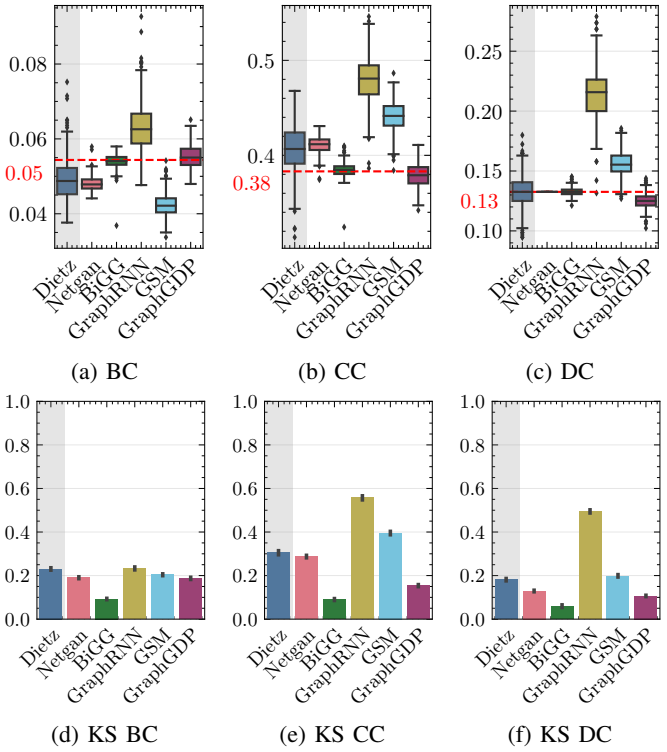
The results for the Géant2001 topology are shown in Figure 4. We can observe that for the graph metrics, Netgan, BiGG, and GraphGDP perform better than the baseline GAN (Dietz) while GraphRNN and GSM perform mostly worse. We note that Netgan always matches the DC perfectly. The Kolmogorov-Smirnov (KS) distance of these metrics exhibits a smiliar pattern. This means that Netgan, BiGG, and GraphGDP outperform the basline in generating (I) graphs with similar mean properties, and (II) graphs in which the properties are similarly distributed throughout the graphs.

The results for the BREN topology are shown in Figure 5. We can observe a slightly different behavior as for the Géant2001 topology. For example, Netgan performs significantly worse on this topology while still maintaining a perfect match of DC. Another example is that GraphGDP performs similar to the baseline on BC but exhibits an extremely low KS distance on BC. This indicates that it is able to distribute the shortest paths that pass through a given node significantly more realisticly in the generated networks. We assume this is of high importance specifically for the BREN topology since it consists of a very densly connected subgraph and a very sparsly connected subgraph. Furthermore, we assume this topologic structure is the reason why all other approaches exhibit similar metrics but significantly worse KS distances compared to the Géant2001 topology.

The results for the BtNorthAmerica topology are shown in Figure 6 We can, again, observe better results for the Netgan, BiGG, and GraphGDP methods while GraphRNN has a hard time accurately generating graphs. BiGG has especially low errors in all distance metrics. We assume these high accuracies are due to the more homogeneous topology structure of BtNorthAmerica.

The results for GtsSlovakia show mixed results (plots omitted for brevity). Netgan performs worse than the baseline on this topology while BiGG performs slightly better and GraphGDP is the only siginificantly better method compared to the baseline. Netgan exhibits especially large errors in the KS distance for CC while BiGG exhibits large errors in the KS distance for DC. We assume that is the case because GtsSlovakia has high variation in the node degrees. Therefore, it is easy to approximate a mean measure over the degrees and harder to distribute them adequately throughout the topology.

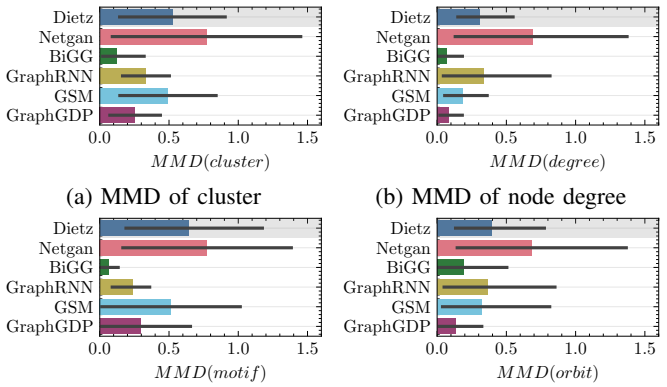
Next, we move from graph metrics and their distribution distances to the Maximum Mean Discrepancy (MMD) over



(a) BC (b) CC (c) DC

(d) KS BC (e) KS CC (f) KS DC

Fig. 6: Synthesis for the BtNorthAmerica topology. Baseline model is highlighted in gray.



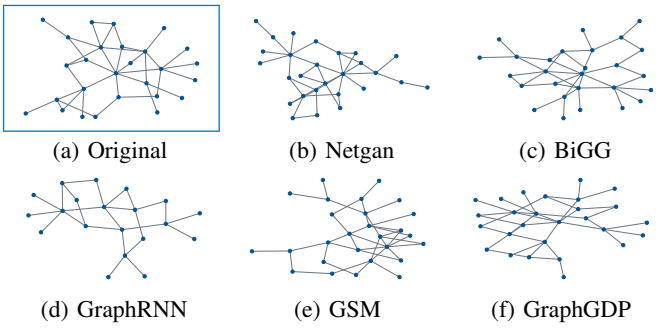
(a) MMD of cluster (b) MMD of node degree

(c) MMD of graph motif (d) MMD of graph orbits

Fig. 7: Maximum Mean Discrepancy (MMD) over various graph and node metrics. Averaged over all topologies. Baseline model is highlighted in gray.

graph metrics. Figure 7 shows the four types of MMD we will be considering. They are averaged over all topologies. We can observe that over all types of MMD, only BiGG and GraphGDP outperform the baseline. GraphRNN and GSM perform similar to the baseline while Netgan performs significantly worse, including an especially large variance.

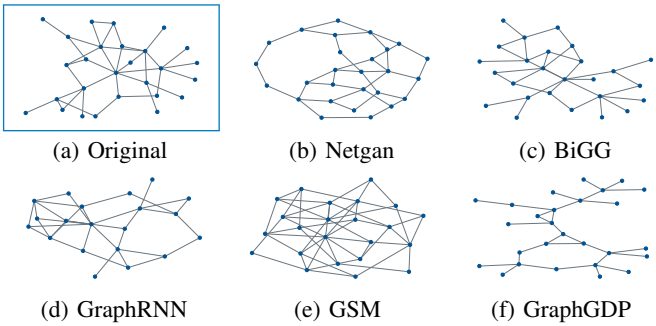
To illustrate the importance of accurately matching these metrics and minimizing the error in KS and MMD, we show generated topologies for each of the methods. We only consider the Géant2001 topology for this purpose. Figure 8 shows for each method a generated topology that achieved a



(a) Original (b) Netgan (c) BiGG

(d) GraphRNN (e) GSM (f) GraphGDP

Fig. 8: Original Géant2001 topology and a generated sample with a low error score for each generation method.



(a) Original (b) Netgan (c) BiGG

(d) GraphRNN (e) GSM (f) GraphGDP

Fig. 9: Original Géant2001 topology and a generated sample with a high error score for each generation method.

low error score. While Figure 9 shows generated topologies with a high error score. We can observe, for example, for Netgan in Figure 8b and Figure 9b, that a low error score leads to a graph with similar structure to the original (Figure 8a) and a high error score leads to a graph where at least parts of the topology are very differently structured.

We summarize the capabilities of the five graph generation methods in comparison to the baseline GAN in Table III. It shows the mean KS distances and MMD values averaged over all topologies per method. We can observe that both BiGG and GraphGDP are strictly better than the baseline for all metrics. Furthermore, BiGG and GraphGDP claim the best performance for all metrics, also against Netgan, GraphRNN, and GSM. BiGG mostly performs well for the MMD metrics while GraphGDP performs best for all KS distances. This highlights the need for a diverse set of metrics to explore capabilities of graph generators.

B. Graph Up-scaling

We evaluate three graph up-scaling tools on the task of scaling Internet Topology Zoo networks to different scale factors. The three approaches are Kronecker, Gscaler, and EvoGraph as listed in Table II. We scale each topology with each method to a scale factor of $2^1 \dots 2^8$.

Figure 10 shows the scaling of the BREN topology which is chosen because of its unique topological structure. The top row shows Kronecker, the middle row Gscaler, and the bottom row EvoGraph. Each row starts with the original graph and progresses from a scale factor of 2 to a scale factor

TABLE III: Mean distance metrics KS and MMD over all topologies per method. Methods with the best performing metric are highlighted.

Method	KS BC	KS CC	KS DC	MMD(degree)	MMD(cluster)	MMD(motif)	MMD(orbit)
Dietz et al.	0.338	0.490	0.234	0.305	0.525	0.639	0.390
Netgan	0.396	0.579	0.274	0.691	0.771	0.775	0.685
BiGG	0.249	0.334	0.197	**0.066**	**0.123**	**0.064**	0.191
GraphRNN	0.238	0.517	0.460	0.339	0.334	0.235	0.363
GSM	0.201	0.431	0.279	0.187	0.493	0.515	0.325
GraphGDP	**0.147**	**0.311**	**0.180**	0.084	0.255	0.293	**0.133**

TABLE IV: Mean distance metrics KS and MMD over all topologies and scales per method. Methods with the best performing metric are highlighted.

Method	KS BC	KS CC	KS DC	MMD(degree)	MMD(cluster)	MMD(motif)	MMD(orbit)
EvoGraph	**0.443**	**0.702**	**0.857**	1.630	**0.935**	1.863	1.930
Gscaler	0.531	0.980	0.886	**1.058**	1.356	**0.972**	**0.984**
Kronecker	0.483	0.714	0.914	1.995	1.253	1.382	1.956

of 64. We can observe that Kronecker populates the graph with new nodes and edges without accurately retaining the topological structure. Gscaler generates two disconnected sub graphs at scale factor 2 and three disconnected sub graphs for each scale factor above that. EvoGraph manages to scale the topology while maintaining the topological structure over all scale factors. We assume this is due to the preferrential edge-attachment mechanism employed by EvoGraph, which keeps the underlying structure of the graph and adds edges in a structure preserving manner in each step.

Table IV shows a summary of all KS distance and MMD metrics for each method averaged over all scales and topologies. The KS distance metrics accurately identify EvoGraph as the most suitable method for WAN topology generation. However, the MMD metrics prefer Gscaler which is not an accurate representation of an intuitively worse-performing approach.

VI. Limitations

While we compare five methods for graph generation with themselves and to a baseline as well as three methods for graph up-scaling, there are approaches that are not covered by this work. We selected the methods to provide a variety of different existing approaches, while not selecting approaches that are very similar to one another.

Furthermore, some approaches are capable of generating and up-scaling graphs, e.g. by setting the scale factor to 1 for generation and any other number for scaling. We don't consider this and divide approaches strictly into generation and up-scaling, based on their main purpose.

Next, we concentrated on undirected graphs without edge weights, since this is the largest common set of supported features of all methods.

Additionally, we don't include any classical graph generation methods in our comparison. We refer to Dietz et al. [2] for a comparison of these approaches.

Lastly, we don't perform any hyperparameter optimization on any of the approaches. Therefore, this comparison serves the purpose of testing the performance of readily available digitial artifacts to be used in research and not necessarily the methodologies themselves.

VII. Reproducibility

We provide access to our data and scripts to obtain this data[1]. More explicitly, we provide the following datasets.

1) Graph generation:
   - 100 synthetic Géant2001 topologies
   - 100 synthetic BREN topologies
   - 100 synthetic BtNorthamerica topologies
   - 100 synthetic GtsSlovakia topologies
2) Graph up-scaling:
   - Géant2001 topologies with scale factors 2-256
   - BREN topologies with scale factors 2-256
   - BtNorthAmerica topologies with scale factors 2-256
   - GtsSlovakia topologies with scale factors 2-256

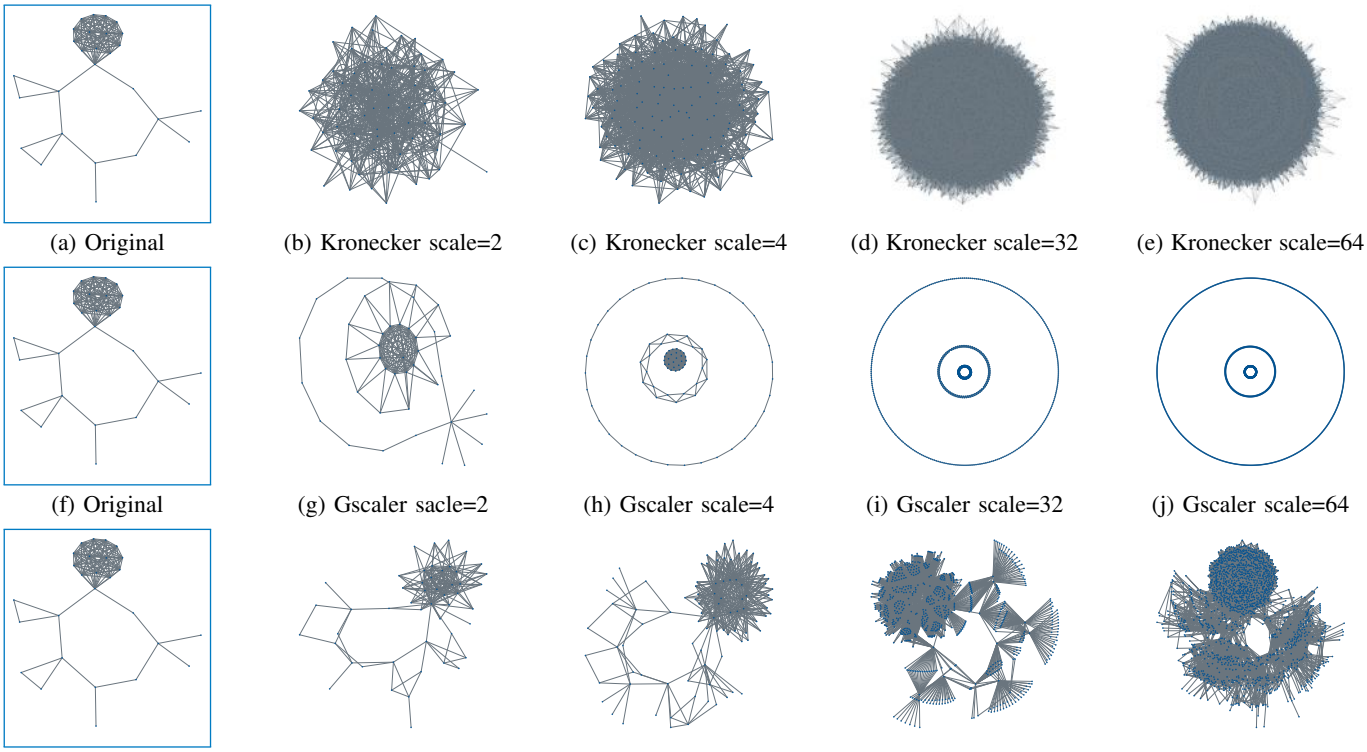
VIII. Conclusion

We performed a comparison of five (mostly) permutation-invariant graph generation methods to a permutation-variant GAN-based baseline and between themselves. We showed that, for the task of WAN topology generation, two approaches outperform the baseline (and current state-of-the-art for WAN topology generation in the literature [2]). We achieved Kolmogorov-Smirnov distances of 0.15, 0.31, and 0.18 compared to the baseline of 0.34, 0.49, and 0.23. This is a percentage improvement of 79.87%. We note that we restricted ourselves to undirected graphs without edge weights.

Furthermore, we performed a comparison of three graph up-scaling methods. We showed that WAN topologies can be scaled with a scaling factor of up to 256 without losing their topological structure when carefully selecting the up-scaling method.

Lastly, we provide a dataset of generated and up-scaled WAN topologies.

[1] https://github.com/tgnn-test/dataset-graphscaling



(a) Original (b) Kronecker scale=2 (c) Kronecker scale=4 (d) Kronecker scale=32 (e) Kronecker scale=64

(f) Original (g) Gscaler sacle=2 (h) Gscaler scale=4 (i) Gscaler scale=32 (j) Gscaler scale=64

(k) Original (l) EvoGraph scale=2 (m) EvoGraph scale=4 (n) EvoGraph scale=32 (o) EvoGraph scale=64

Fig. 10: Graph up-scaling at the example of the BREN topology. The first row uses Kronecker, ther second row Gscaler, and the last row EvoGraph. We show the evolution from the original graph (left) over scales 2, 4, 32, and 64 (right).

ACKNOWLEDGMENT

European Union Horizon 2020 (project SLICES-SC, 101008468, and SLICES-PP, 101079774). Bavarian Ministry of Economic Affairs, Regional Development and Energy (project 6G Future Lab Bavaria). German Federal Ministry of Education and Research (project 6G-life, 16KISK002, and project 6G-ANNA, 16KISK107).

REFERENCES

[1] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[2] K. Dietz, M. Seufert, and T. Hoßfeld, "Comparing Traditional and GAN-based Approaches for the Synthesis of Wide Area Network Topologies," in *2022 18th International Conference on Network and Service Management (CNSM)*. IEEE, 2022, pp. 64–72.

[3] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "Netgan: Generating Graphs via Random Walks," in *International conference on machine learning*. PMLR, 2018, pp. 610–619.

[4] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "GraphRNN: Generating Realistic Graphs with Deep Auto-Regressive Models," in *International conference on machine learning*. PMLR, 2018.

[5] C. Niu, Y. Song, J. Song, S. Zhao, A. Grover, and S. Ermon, "Permutation Invariant Graph Generation via Score-based Generative Modeling," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 4474–4484.

[6] H. Dai, A. Nazi, Y. Li, B. Dai, and D. Schuurmans, "Scalable Deep Generative Modeling for Sparse Graphs," in *International conference on machine learning*. PMLR, 2020, pp. 2302–2312.

[7] H. Huang, L. Sun, B. Du, Y. Fu, and W. Lv, "GraphGDP: Generative Diffusion Processes for Permutation Invariant Graph Generation," in *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2022, pp. 201–210.

[8] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routenet: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2260–2270, 2020.

[9] M. Ferriol-Galms, J. Paillisse, J. Surez-Varela, K. Rusek, S. Xiao, X. Shi, X. Cheng, P. Barlet-Ros, and A. Cabellos-Aparicio, "RouteNet-Fermi: Network Modeling with Graph Neural Networks," 2022. [Online]. Available: https://arxiv.org/abs/2212.12070

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[11] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[12] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics," 2015.

[13] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An Approach to Modeling Networks." *Journal of Machine Learning Research*, vol. 11, no. 2, 2010.

[14] P. ERDdS and A. R&wi, "On Random Graphs I," *Publ. math. debrecen*, vol. 6, no. 290-297, p. 18, 1959.

[15] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[16] D. J. Watts and S. H. Strogatz, "Collective Dynamics of Small-world Networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[17] P. Holme and B. J. Kim, "Growing Scale-free Networks with Tunable Clustering," *Physical review E*, vol. 65, no. 2, p. 026107, 2002.

[18] J. Zhang and Y. Tay, "GSCALER: Synthetically Scaling A Given Graph." in *EDBT*, vol. 16, 2016, pp. 53–64.

[19] H. Park and M.-S. Kim, "EvoGraph: An Effective and Efficient Graph Upscaling Method for Preserving Graph Properties," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2051–2059.