Want More WANs? Comparison of Traditional and GAN-Based Generation of Wide Area Network Topologies via Graph and Performance Metrics

Katharina Dietz, *Student Member, IEEE*, Michael Seufert, *Member, IEEE*, and Tobias Hoßfeld, *Senior Member, IEEE*

***Abstract*—Wide Area Network (WAN) research benefits from the availability of realistic network topologies, e.g., as input to simulations, emulators, or testbeds. With the rise of Machine Learning (ML) and particularly Deep Learning (DL) methods, this demand for topologies, which can be used as training data, is greater than ever. However, public datasets are limited, thus, it is promising to generate synthetic graphs with realistic properties based on real topologies for the augmentation of existing data sets. As the generation of synthetic graphs has been in the focus of researchers of various application fields since several decades, we have a variety of traditional model-dependent and model-independent graph generators at hand, as well as DL-based approaches, such as Generative Adversarial Networks (GANs). In this work, we adapt and evaluate these existing generators for the WAN use case, i.e., for generating synthetic WANs with realistic geographical distances between nodes. We investigate two approaches to improve edge weight assignments: a hierarchical graph synthesis approach, which divides the synthesis into local clusters, as well as sophisticated attributed sampling. Finally, we compare the similarity of synthetic and real WAN topologies and discuss the suitability of the generators for data augmentation in the WAN use case. For this, we utilize theoretical graph metrics, as well as practical, communication network-centric performance metrics, obtained via OMNeT++ simulation.**

***Index Terms*—Wide area networks (WAN), generative adversarial networks (GAN), graph generation, network performance, network simulation.**

I. Introduction

GRAPHS are an important concept in many research areas, and their topology often has decisive impact on the studied systems, whether it is the communication flow in social networks, the composition of chemical molecules, or the architecture of computer networks. For the latter, the network topology has a big impact on research for Wide Area Networks (WAN), such as the controller placement in Software-defined Networking (SDN), the gateway placement in Long Range Wide Area Networks (LoRaWANs), or the prediction of Key Performance Indicators (KPIs), such as round-trip times (RTTs) and network load. Researchers often resort to testbeds, simulations, or emulators for parameter studies, and thus, require realistic network topologies to obtain meaningful results.

Several public datasets exist, such as the Internet Topology Zoo (ITZ) [1] or the Survivable fixed telecommunication Network Design library (SNDlib) [2], containing over 250 and 25 different network topologies, respectively. However, as the size of these datasets is limited, researchers already have expressed the concern that the zoo is too small for their field of application [3], and resorted to simple algorithmic approaches to create new data points [4].

Another possibility to obtain realistic network topologies is to use model-based graph generators, such as Barabási–Albert (BA) [5], Erdős–Rényi (ER) [6], and Watts-Strogatz (WS) [7], which can produce topologies with desired properties. In combination with small sets of realistic network topologies, the addition of synthetic network topologies with realistic properties allows for data augmentation, which is especially beneficial for Machine Learning (ML)-based approaches [8]. Besides data augmentation, the synthesis of network graphs may also serve as a privacy mechanism, as organizations or enterprises may be hesitant to publish information about their network topology, as tomography-based topology inference poses a crucial threat to communication networks [9]. Still, these algorithmic approaches may not always yield optimal results and show room for improvement [4], as the layout of computer networks does not adhere to any general model [1].

However, in recent years, more sophisticated and model-agnostic ways to synthesize networks have arisen. Instead of conforming to an underlying model, these approaches take information from the real network as input, e.g., the joint degree distribution (JDD) [10], and try to replicate the real network as close as possible. Similarly, Deep Learning (DL)-based approaches such as Generative Adversarial Networks

Manuscript received 17 March 2023; revised 7 July 2023; accepted 7 July 2023. Date of publication 24 July 2023; date of current version 7 February 2024. This work was partly funded by the German Federal Ministry of Education and Research (BMBF) as part of the project WINTERMUTE (16KIS1129), partly funded by Deutsche Forschungsgemeinschaft (DFG) under grant SE 3163/3-1, project number: 500105691 (UserNet), and in part by the Open Access Publication Fund of the University of Würzburg. The associate editor coordinating the review of this article and approving it for publication was S. Kanhere. *(Corresponding author: Katharina Dietz.)*

The authors are with the Chair of Communication Networks, University of Würzburg, 97070 Würzburg, Germany (e-mail: katharina.dietz@uni-wuerzburg.de; michael.seufert@uni-wuerzburg.de; tobias.hossfeld@uni-wuerzburg.de).

Digital Object Identifier 10.1109/TNSM.2023.3298205

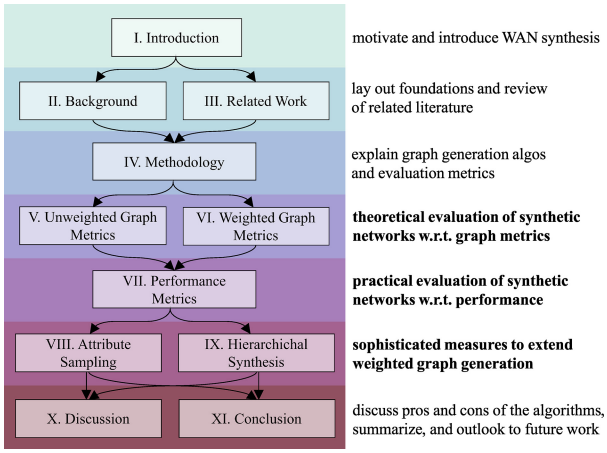


Fig. 1. Graphical abstract of this work, **highlighted** chapters depict the main part (shortened alternative chapter names).

(GANs) [11] are also model-agnostic. They use presented information and obfuscate it to synthesize new data.

Summarizing, research on graph generation presents us with model-dependent, model-independent, and DL-based graph generation. However, it is unclear, which graph generation is best suited for WAN research. The complexity of WAN generation lies in the geographical distances of links, as these distances directly influence a network's performance, and the weights are not easily interchangeable, i.e., we cannot switch the weight of a link connecting two continents with the weight of an edge connecting two cities, as it would drastically influence graph metrics and consequently also performance metrics, such as the RTT. Thus, the goal of this paper is to evaluate the vastly different approaches with respect to the WAN use case and discuss their pros and cons. To the best of our knowledge, there exists no work on synthesizing WANs by adopting and comparing traditional as well as DL approaches.

The main contributions of this paper are:

1) We evaluate graph generation approaches for the WAN problem, comparing the similarity of synthetic and real network topologies via theoretical graph metrics as well as practical performance metrics.
2) We study both unweighted and weighted graphs, for which the latter contain additional link attributes, such as the geographical distance of the WAN nodes.
3) We investigate two approaches to improve edge weight assignments: a hierarchical graph synthesis approach, which divides the synthesis into local clusters, as well as sophisticated attributed sampling.
4) We publish our code and generated networks [12], [13].

The remainder of this work is structured as depicted in Figure 1. Firstly, in Section II, we give background information about network topologies, algorithmic graph generators, GANs, and graph/performance metrics, while Section III outlines related work. In Section IV, we specify the methodology, e.g., utilized networks, graph generators, and metrics. We then evaluate the generators' capability to synthesize unweighted network topologies similar to the real networks in Section V, whereas Section VI adapts the initial approach to generate weighted graphs, representing WANs. In Section VII we simulate the networks to analyze load- and delay-based performance metrics. Sections VIII and IX investigate approaches to improve edge weight assignments: sophisticated attribute sampling and a hierarchical graph synthesis approach. Section X highlights the respective (dis)advantages of the studied approaches. Lastly, we summarize our findings and provide an outlook for future work in Section XI.

This work is an extended version of our recently published paper [14]. In detail, we add an evaluation of more practical load- and delay-based performance metrics in addition to the theoretical graph metrics for both existing approaches (naïve, hierarchical). In addition to the two existing approaches, we also investigate a completely new approach to improve the assignment of weighted edges: sophisticated attribute sampling. To analyze the performance metrics, we simulate the networks via OMNeT++ to mimic a software-defined WAN (SD-WAN), resulting in 148,000 simulation runs, equating to 2.8 years of real-world time. Last but not least, we extend on related work and background information as well as adding a more detailed insight into to the hierarchical approach.

TABLE I
UTILIZED NOTATIONS, ADAPTED FROM FAEZ ET AL. [16]

Notation	Description
V	Node (or vertex) set of a graph.
E	Edge (or link) set of a graph.
G	A graph, $G = (V, E)$.
n	Number of nodes, $n = \|V\|$
m	Number of edges, $m = \|E\|$
$N(v)$	Neighbour set of a node $v \in V$
$e(v_1, v_2)$	The edge e connecting the node v_1 to a node v_2.
$sp_u(v_1, v_2)$	Shortest unweighted path from a node v_1 to a node v_2.
$sp_w(v_1, v_2)$	Shortest weighted path from a node v_1 to a node v_2.
$deg(v)$	Degree of a node v.

II. BACKGROUND

In this section, we briefly outline characteristics of WANs. Furthermore, we discuss the possibility of creating synthetic networks via algorithmic and DL-based approaches. Lastly, we discuss graph and performance metrics, which may be used for graph/network analysis.

WANs: Computer networks can be generally divided into different categories, e.g., regarding their geographical distribution and functionality. The main characteristic of WANs compared to other computer networks are long(er) links between the comprising nodes, e.g., intercontinental or cross-country connections, having great impact on the performance metrics like the RTT. WAN generation is more complex compared to computer networks where there is no such big variance, as a misplacement of such a long link may have great impact on the metrics. As identified by Knight et al. [1] the layout of such computer networks is not conforming to any common guidelines, thus, showing great diversity in their design and differing greatly from other networks, such as social networks [15]. We interpret a WAN as a Graph G, where intermediary devices such as routers and switches depict the nodes V, and links between those devices depict the edges E. Table I illustrates common notations used in this paper.

Algorithmic Graph Generators: As available data for WAN topologies is exhaustible, generating more networks is desirable. Model-dependent graph generators typically have one or more input parameter via which the model generation can be controlled, and do not need any information about the real network, that we are trying to replicate. Nevertheless, the parameters need to be configured properly and may not be universally applicable. Thus, throughout the years, more sophisticated, model-agnostic approaches emerged. Instead of being configurable via input parameters, the generators take explicit information about the network we are trying to replicate, which they then try to match as close as possible. This type of generation is often coupled with graph sampling to approximate characteristics of larger networks.

GANs: With the rise of DL, approaches such as GANs [11] – widely used for image generation – have also been adapted for many other applications such as topology generation. Generally, a GAN consists of two neural networks (NNs), namely the discriminator and the generator. The generator is responsible for synthesizing fake data samples and thus tries to trick the discriminator. It starts off with random noise and then iteratively improves its output, whereas the discriminator is fed real data as well as the synthesized samples from the generator and tries to classify them correctly. GANs can be seen as a zero-sum game between those two actors, where the discriminator tries to minimize the so-called loss, which represents the discriminator's ability to distinguish between real and fake data. The generator tries to maximize this loss. By simply taking a graph's adjacency matrix as input for the GAN, we can easily utilize existing approaches for GAN-based data generation, as both images and adjacency matrices are also basically just $n \times n$ matrices.

Graph Metrics: Many graph metrics can be calculated and compared between real and synthesized networks. Usually, most graph metrics are based on the shortest paths in the network or the node degree [17], and may be node- or graph-based, i.e., consist of multiple values for all nodes, or just one for the whole graph. The closer the metrics for the synthesized samples are, the better the reconstruction is. However, as already observed by Bojchevski et al. [18], perfectly replicating a network is not the goal, as otherwise one could just copy the existing data point in the dataset, without performing any sort of network synthesis. Thus, while we want the generated networks on average to have similar graph metrics to the real network, variations in the metrics are still desired.

Performance Metrics: The aforementioned static graph metrics compare the generated topologies on a more theoretical level. Thus, it is unclear how such a synthesized WAN would perform on a more practical level with regards to performance metrics, which may potentially be more complex as they model dynamic processes [17]. In communication networks, many performance metrics are delay- or workload-based, e.g., RTT and throughput, respectively. Performance and graph metrics often correlate, e.g., the RTT may increase if there are more nodes between sender and receiver. Consequentially, while similar graph metrics can potentially depict a decent base for synthesized networks, it is important to also evaluate their performance for a specific use case.

SDN: In this paper, we evaluate the performance metrics of the synthetic networks for the use case of SDN, more precisely, SD-WANs. Due to the ever increasing complexity of communication networks, SDN emerged as a design paradigm to increase the flexibility and manageability of communication systems [19], [20] by separating the control plane from the data plane. Here, the network's intelligence resides in a central controlling entity, i.e., the SDN controller, which may pose a scalability and elasticity issue due to being a potential performance bottleneck and single point of failure. Thus, evaluating more SDN-specific performance metrics such as the controller load is of great interest. However, access to large scale SDN-enabled networks, including compatible hardware, is limited [21], [22]. Frameworks such as the OpenFlow OMNeT++ Suite (OOS) [23] have emerged, enabling simulation of a variety of network topologies in an SDN context.

III. Related Work

In this section, we give an overview of related work concerning graph generation, and discuss which approaches we adopt for the WAN synthesis.

Algorithmic Graph Generation: First approaches to synthesize networks via algorithmic generators date back decades. One of the most well-known and simplest approach is the Erdős-Rényi (ER) model [6], where a graph is simply constructed by defining a probability, that an edge between two nodes is created. As the rather random structure of ER graphs may not be fitting, alternatives are the Barabási-Albert (BA) model [5], which models a preferential attachment of nodes by creating *hubs*, or the Watts-Strogatz (WS) model [7], which starts off with a (chord) ring structure and then randomly rewires edges. Clearly, all of these models make underlying assumptions about the graphs, which may only be applicable in specific cases. We decide to evaluate ER, BA, and WS models, as they are cheap and simple models, can potentially pose a very easy solution to our problems, and are still prevalent in current literature [24].

Research on graph synthesis nowadays has shifted to a more model-agnostic way of modeling graphs. The 2K+-framework [10] is a state-of-the-art approach of such a model-agnostic approach. Given explicit information such as a joint degree matrix (JDM), the algorithm tries to construct a graph that matches this information as close as possible. We decide to pick 2K graphs as well, as they depict an up-to-date approach, that has been constantly extended in recent years. It adds a more complex/sophisticated way to generate graphs compared to the previous, model-based approach.

The above approaches are mainly geared towards social networks and have been tested thoroughly in this field of application. The goal of this work is to investigate these approaches in the context of WANs by providing an extensive analysis of weighted and unweighted graph metrics.

Deep Graph Generation: An alternative to algorithmic approaches is DL-based graph generation. Generally, DL-based generative models can be divided into two types, namely

implicit and explicit models [25]. The latter directly samples/learns from the (estimated) probability distribution, while the former *learns* how to sample from the distribution, without directly modeling it [16], [25]. Popular examples for the two approaches are GANs and Variational Autoencoders (VAEs) [26], respectively. In this work, we decide to investigate the more implicit approach of GAN-based topology generation, as it complements the more explicit approach of the previous 2K algorithm. This DL-based graph generation depicts the most complex, but also most flexible approach of our comparison.

As showcased by Faez et al. [16], research for such deep graph generators mainly focuses on chemical and bioinformatics [27], [28], social structures [18], [29], [30], [31], or synthetic datasets [30], [31], created by ER, WS, and others. In this paper, we adopt some techniques from other application fields to evaluate for our use case of WAN generation. Specifically, we take inspiration from Tavakoli et al. [32] for the general approach, and Liu et al. [33] for the hierarchical synthesis. Though, our methodology drastically varies in terms of utilized GAN architecture or utilized clustering/community detection algorithms in the hierarchical case. Both works focus on social and/or synthetic datasets. WANs differ from networks in other research areas, as the works mainly zero in on unweighted graphs (possibly associated categorical node attributes). To the best of our knowledge, there exists no work on synthesizing WANs via DL-based approaches, as well as a comparison with traditional approaches.

Graph vs Performance Metrics: As the aforementioned literature does not zero in on synthesizing communication networks, they also draw no comparison with regards to network performance between real and synthetic networks, and consequently do not relate graph and performance metrics. However, other works illustrate the importance of this relationship, e.g., relating the network topology to SDN performance prediction [4], [34], TSN latency assignments [35], network resilience [36], or Quality of Service (QoS) [37]. Thus, in this paper, we also investigate performance metrics of the synthetic networks in addition to the graph metrics to quantify the quality of the network generation.

IV. Methodology

In this section, we firstly outline the chosen networks we utilize in our studies. Secondly, we explain the chosen network generators and their configuration. We proceed by defining the graph metrics we use for comparing synthetic networks to the real network. Lastly, we conclude this section by explaining the simulation setup and performance metrics.

A. Chosen Networks

We pick four networks from the ITZ, which are depicted in Figure 2, as they cover topologies with varying characteristics. In detail, we utilize a subset of the ITZ provided by Gray et al. [34], [38] where networks from the original ITZ in *.graphml* format have been transformed into a more readable format, e.g., given longitudes and latitudes have already been converted to physical distances, hyper-edges have been resolved, or nodes with no location or connection have been removed. Thus, any description of these networks adheres to the conversion of Gray et al. Note that some connections initially have a weight of zero, which we set to one (minimum weight), to make them distinguishable from non-connections.

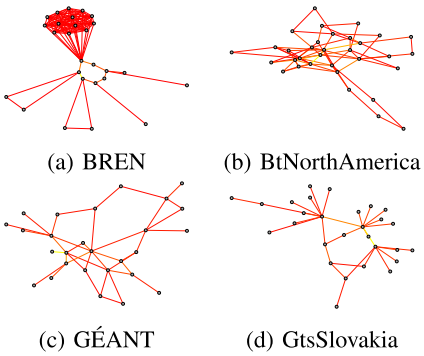


Fig. 2. The chosen WANs from the ITZ – red links (—) depict small geographical distances (in relation to the respective maximum distance), yellow links (—) bigger ones.

1) BREN: The first network is the Bulgarian Research and Education Network (BREN), connecting various Bulgarian universities, representing the network in the ITZ subset with the largest number of links, with $m = 107$ and $n = 27$. It has a peculiar layout, with inter-meshed clusters, and thus is particularly interesting to investigate. Again, we want so emphasize on the importance of geographical distances of links in a WAN, as the (in)correct inference of those weights may drastically influence a network's performance. Placing one of the longer links in the fully-meshed clusters here will skew the average path delay.

2) BtNorthAmerica: Next, we also choose the network with the second most links, drastically decreased to $m = 70$ and $n = 33$. It depicts a network distributed throughout the whole continent of North America, thus spanning a wider geographical range than BREN, while also being structurally different with a more random layout.

3) GÉANT: The third network we choose is GÉANT as it represents a medium network with a close to median amount of links with $m = 38$ and $n = 27$ (median is 38.5 links). GÉANT is the collaboration of various European National Research and Education Networks (NRENs), connecting them, including the previously introduced BREN.

4) GtsSlovakia: Lastly, we choose an even sparser network as our fourth option, namely GtsSlovakia with $m = 30$ and $n = 28$. Note that we specifically do not opt for the network with minimum links here, which would be GtsCzechRepublic, as this network is mostly path-like. Instead we choose a network with a more characteristic network design, as it fits the general model idea of one of the traditional generators, and we find that an in-depth discussion is more appropriate. We will, however, briefly discuss the results for GtsCzechRepublic in the evaluation as well for the sake of consistency.

B. Network Generators

Before synthesizing networks, we have to decide on a set of network generators and configure them in a fair way. In the

following, we adhere to the nomenclature of NetworkX [41], which we use for the algorithmic generators.

1) ER: To configure the ER-model, we may only influence the output via the p_{ER} parameter, depicting the probability that a specific edge is constructed between two nodes. Thus, the number of expected edges is $\binom{n}{2} \cdot p_{ER}$ edges, as in a fully connected graph there are $\binom{n}{2} = \frac{n(n-1)}{2}$ edges. Therefore, we configure p_{ER} as follows to match a network (on average) with *m* edges: $p_{ER} = \frac{2m}{n(n-1)}$. As it may occur that the generated graph is disconnected, we rewire the graph by adding a random edge between the largest component and smaller components/isolated nodes, and then capping an edge in the graph for every added edge while ensuring connectivity.

2) BA: To configure the BA-model, we may only change the output via the m_{BA} parameter, representing the number of other nodes a newly added node is connected to, when generating the network. On average, a node in the BA-model has a degree of $2 \cdot m_{BA}$ [42, p. 432], i.e., $n \cdot m_{BA}$ expected edges for an undirected graph. Therefore, we configure m_{BA} to match a network (as close as possible) with *m* edges as follows: $m_{BA} = \lfloor \frac{m}{n} \rceil$. Since m_{BA} has to be an integer, we round it to the nearest integer. Here, the output is always connected, thus no need for rewiring.

3) WS: To configure the WS-model, we may change two parameters, i.e., the number of nearest-neighbours k_{WS} a node initially connects to, and the probability p_{WS} that an initial edge is rewired. While the parameter k_{WS} actually influences the total number of edges, p_{WS} only affects the rewiring. As each node is connected to its k_{WS} closest neighbours, the average node degree is also k_{WS}, meaning $\frac{k_{WS} \cdot n}{2}$ edges in the network. To model a network with close complexity to a network with *m* edges, we configure: $k_{WS} = \lfloor \frac{2m}{n} \rceil$.

As the node is connected to k_{WS} nodes only if k_{WS} is even, and $k_{WS}-1$ if it is odd, we also distinguish between the calculated k_{WS} and an alternative $k^*_{WS} = k_{WS}+1$ and choose the better fit of both. Lastly, the higher p_{WS} is configured, the more chaotic the graph gets, so we set $p_{WS} = 0.2$, to retain the possible ring-like structure, as a totally randomly generated graph is already covered by the ER-model. Similarly to ER, we will rewire the graph in the same manner, if the output is not fully connected.

4) 2K: Compared to the three previous classical model-based graph generators, 2K-graphs [10] are a state-of-the-art approach for graph synthesis. Instead of adhering to a general underlying model that we need to configure, the algorithm takes the joint-degree distribution (JDD) of the graph as input. In detail, the algorithm takes the joint degree matrix (JDM) as input, where the element in the *i*-th row and *j*-th column depicts the number of nodes of degree *i* attached to nodes of degree *j*. The core idea of the algorithm is to start with each node having *stubs* which edges can be connected to, reflecting the degree of that node. Edges are then iteratively added to the graph to match the given JDD. Thus, instead of making assumptions about some sort of graph model, 2K directly infers information about the network to be generated, in this case the JDD. Again, we will rewire the graph as before, if the output is not fully connected.

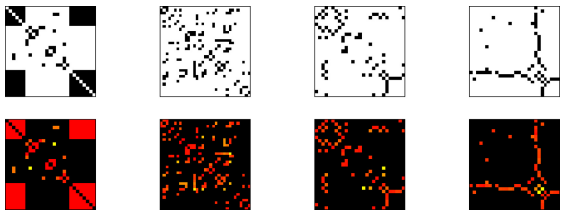


Fig. 3. BREN, BtNorthAmerica, GÉANT, and GtsSlovakia as images (left to right), top: as BW, bottom: as RGB.

5) GAN: In the following, we explain the setup of the GAN-based approach. As this is more complex, we subdivide this into three parts, namely input, architecture, and output.

Input: As mentioned in the previous section, we utilize a network's unweighted and weighted adjacency matrix for the GAN approach, which we can interpret as images as illustrated in Figure 3. The top row illustrates the four networks from Figure 2 as simple black-and-white (BW) images, where a black pixel indicates that there is a link between nodes. Similarly, the bottom row depicts the topologies as red-green-blue (RGB) images, where a black pixel with RGB-channels (0,0,0) depicts no connection, and the gradient from yellow to red depicts a connection with varying min/max-normalized geographical distances and analogous color-coding to the networks in Figure 2, i.e., RGB-color channels of (1, *i*, 0) with $i \in [0, 1]$. Similar GAN input has been proven useful in other use cases, such as dynamic link prediction [43]. Though, we deliberately utilize a second color channel instead of utilizing the full spectrum of the greyscale, as otherwise a lower value would indicate a less important link, but in our scenario every link is equally important, independently from the geographical distance. In other words, the red color channel is a binary value, which simply states the existence of a link, whereas the green color channel is a continuous value, depicting the geographical distance. We do not utilize the blue color channel of images and thus omit it. In the future, this channel may be used to encode bandwidths or other WAN properties. Also, note that we are not restricted to only three color channels, though we choose this analogy for the sake of explainability.

As proposed by Tavakoli et al. [32], we feed the GAN 10,000 permutation matrices of the original network. In other words, we randomly relabel the node ordering, which results in different adjacency matrices/images, while still representing the same network. Thus, we are able to generate new networks with only one original network.

Architecture: Figure 4 illustrates the architecture of our utilized GAN. The architecture is adapted from [39], designed to synthesize images of numbers from 0 to 9, and implemented in TensorFlow. As the image size and overall task complexity is similar to our use case, we reuse this architecture and adapt it. In detail, we parameterize the architecture to take any size of input, as well as adding color-channels, as the initial architecture is only designed for BW images. Additionally, we make the discriminator and generator symmetrical, which is not the case in the original, where the generator has an extra convolutional layer. For BW images, we remove this additional layer from the generator, and for RGB images we add

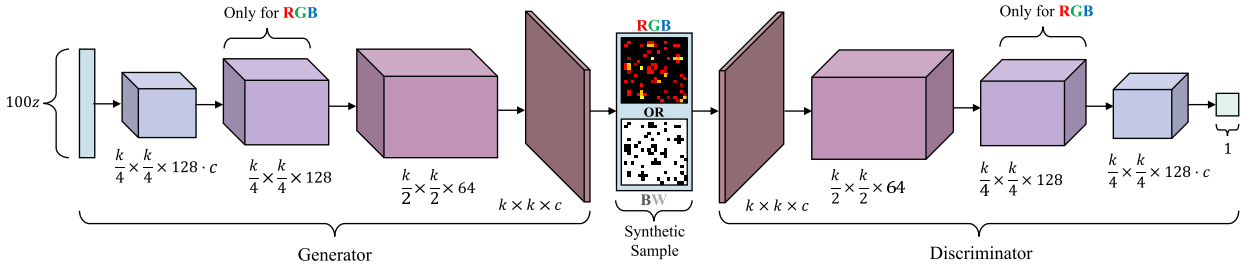


Fig. 4. Architecture of the utilized Deep Convolutional GAN (DCGAN) [39], [40].

an additional layer to the discriminator as well, as BW images are less complex than RGB images.

Output: As the GAN may produce samples, where the entry in the matrix is not exactly binary, i.e., to represent whether an edge exists between two nodes or not, but is a real number between 0 and 1, we need to postprocess the output. For this, we model the probability $p(i, j)$ that there is an edge $e(i, j)$ in the postprocessed adjacency matrix, i.e., $post[i][j] = 1$, as a Bernoulli distribution with $p(i, j) = \frac{pre[i][j]+pre[j][i]}{2}$ depending on $pre[i][j]$, which is the corresponding entry of the preliminary adjacency matrix that was synthesized by the GAN. In other words, if the GAN is confident, that $e(i, j)$ exists, the edge is more likely to persist through the postprocessing. Note that the adjacency matrix for undirected graphs is symmetrical with respect to the diagonal axis, thus we take into account two entries in the above calculation. If we synthesize weighted topologies and have decided on the existence of a link via the first color channel, we choose as link weight the average of the two symmetrical entries of the raw GAN output in the second color channel. Lastly, as for the previous algorithms, the GAN can also produce disconnected graphs, so we again proceed in the same manner.

C. Graph Metrics

To compare the synthesized networks to the real networks, we need to decide on a set of graph metrics. For this, we choose graph metrics that have proven to be useful in the context of communication networks, such as Time-Sensitive Networking (TSN) [35], SDN-enabled performance prediction [4], LoRaWAN gateway placement [44], SDN controller placement [45], or Virtualized Network Function (VNF) placement [46]. Thus, our graph metrics consist of the Betweenness Centrality (BC), Closeness Centrality (CC), and Degree Centrality (DC). For all metrics, we compute the unweighted version, as well as their weighted version with respect to the geographical distances.

The (unweighted) BC of a node is the fraction of shortest paths, that a node is contained in, and is computed as follows:

$$BC_u(v_1) = \sum_{v_1 \neq v_2 \neq v_3} \frac{|\{sp_u(v_2, v_3) | v_1 \in sp_u(v_2, v_3)\}|}{|sp_u(v_2, v_3)|}$$

The weighted BC is computed analogously, with the weighted shortest paths $sp_w(v_2, v_3)$ as basis. The (unweighted) CC of node is the reciprocal of the sum of the shortest path lengths to all other nodes, and is computed as follows:

$$CC_u(v_1) = \frac{|V|}{\sum_{v_2 \in V} |sp_u(v_1, v_2)|}$$

Again, the weighted CC is computed analogously, with the weighted shortest paths $sp_w(v_1, v_2)$ as basis. Lastly, the (unweighted) DC is the (normalized) degree of a node:

$$DC_u(v_1) = \frac{deg(v_1)}{|V| - 1}$$

As there is no universally accepted definition of a weighted DC, we define it as the sum of all outgoing edge weights:

$$Dist_\Sigma(v_1) = DC_w(v_1) = \sum_{v_2 \in V} w(e(v_1, v_2))$$

D. Network Simulation

After specifying the topological metrics which are solely based on the network layout, we also collect performance metrics of SD-WANs. For this, we resort to a simulation-based approach and utilize the OMNeT++-based OOS.

OOS: The OOS has been a popular tool in the past and recent target for extensions for simulating SDN-enabled networks for use cases such as performance prediction [4], [34] or intrusion detection [47]. At its core, the OOS provides us with an implementation of OpenFlow-enabled [48] controllers and switches. As described in the original OOS paper [23] and its implementation [49], the OOS simulates the NOX-MT [50] controller, a multi-threaded extension of the NOX controller [51]. The SDN controller is equipped with a realistic set of functionality, including topology discovery via Link Layer Discovery Protocol (LLDP) messages, forwarding mechanisms, and Address Resolution Protocol (ARP) proxy. Adding to the traffic caused by the aforementioned controller applications, the main traffic source are end-devices/hosts connected to the switches, based on the *StandardHost* from the INET [52] framework in conjunction with a modified version of the *PingApp*, also from INET. To ensure statistical robustness, each network (generated and real) is simulated four times, each repetition reflecting a 10 min timespan.

Performance Metrics: In contrast to the graph metrics, performance metrics are collected during operation/after deployment of the network. As many popular performance metrics are either load- or delay-based, we pick a representative of both categories. The first performance metric we utilize is the load of the SDN controller. The controller load is calculated by the OOS in $\mathrm{B\,s^{-1}}$ on a timeslot-basis at a granularity of 1 s.

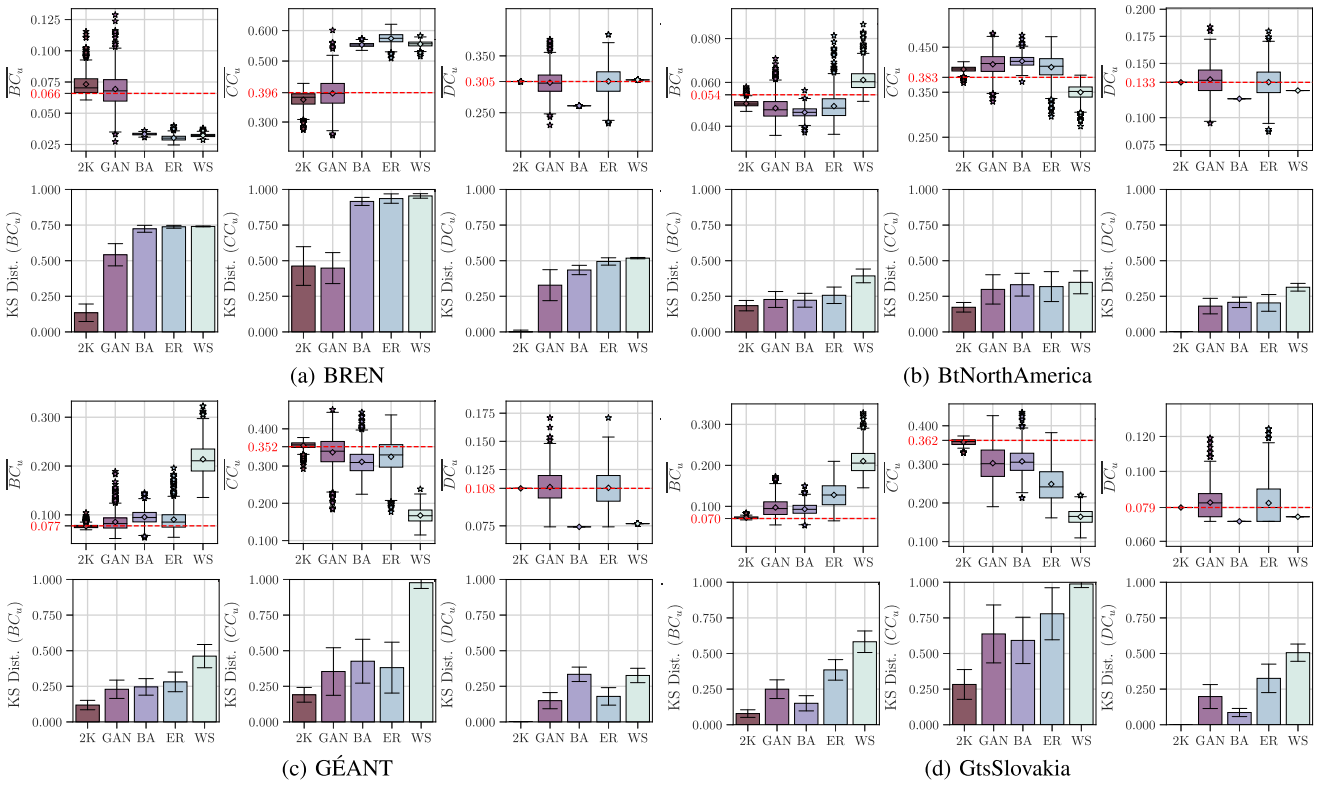


Fig. 5. Unweighted synthesis for the four networks; red dashed line (---) depicts the metrics of the corresponding real network.

The second performance metric we evaluate is the RTT, taking into account various latencies such as queuing and service times at the switches and controllers, the possible stalling when waiting for a controller response, and transmission and propagation delays between nodes. The INET framework equips the user with a tool to easily record the RTT by providing the mentioned *PingApp* module.

V. Synthesis of Unweighted Topologies

In this section, we examine a total of 1,000 synthesized networks for each of the five generators and compare the chosen graph metrics to the real networks. We start off by providing an objective description of the results, and conclude this section with a subjective interpretation. Note that for the GAN, as we sample random permutation matrices and generally GANs tend to oscillate [53], we generate these 1k networks from ten different seeds to obtain robust results.

1) BREN: Figure 5(a) depicts the results for the BREN network. On top, the average BC, CC, and DC of the synthetic BREN-like networks are shown, with the red dashed line showing the real value of BREN. On bottom, the average Kolmogorov-Smirnov (KS) distance between the distribution of the metric in the synthetic networks and in the real network is illustrated. The whiskers indicate the standard deviation of the KS metric. For the average BC and CC, it becomes apparent that the legacy generators are not able to capture the average centrality with their generated networks. This is due to the peculiar structure of BREN, as it contains one big cluster of fully meshed nodes, as well as some smaller clusters. The DC, however, is captured on average perfectly for WS and ER. However, this is expected, as we optimized all generators to have the same, or as similar as possible, number of links as the real network. 2K and the GAN are able to capture all three centrality metrics for the average generated network. The main difference between both here is the variance of generated networks, which is especially obvious for the DC. Looking at the more distribution-focused KS distances, the higher variance of the generated networks of the GAN is also reflected there, with 2K possessing the best fits on average, for the BC and DC, and CC being similar to that of the GAN.

2) BtNorthAmerica: Figure 5(b) depicts the results for BtNorthAmerica in the same fashion as before. Generally, all generators perform very similar – even the legacy ones – as the network is less structured and simply capturing a fitting number of links is more sufficient compared to BREN. However, none of the generators is quite able to catch the average BC and CC and all are slightly off. Again, 2K expresses low variance in the generated networks, especially for the DC, whereas GAN and ER are more deviating from the mean. In this scenario, the GAN acts very similar to ER in terms of average performance and induced variance.

3) GÉANT: Figure 5(c) depicts the results for GÉANT in the same fashion as before. 2K produces again the best fitting networks on average, however, with very low variances. Similar to BtNorthAmerica, GAN and ER perform similarly, as GÉANT, too, is a more random network. Thus, BA and WS do not capture the three centrality metrics, especially obvious for the DC. A look at the KS distances confirms this, which also shows that GAN has a slight advantage over ER.

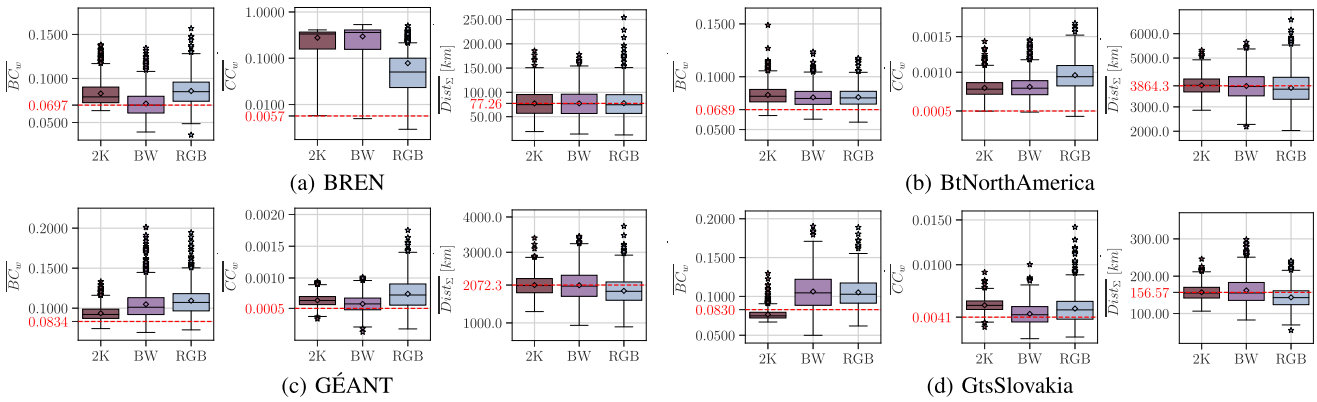


Fig. 6. Weighted synthesis for the four networks; red dashed line (---) depicts the metrics of the corresponding real network.

4) GtsSlovakia: Figure 5(d) depicts the results for GtsSlovakia in the same fashion as before. Again, 2K creates networks close the original network, but with very low variance. BA performs second best here, as the star-shaped topology adheres to the underlying model of BA well. It becomes also apparent, that GAN is better at approximating the real network than a random generator, i.e., the ER-model, thus depicting a definite advantage over the traditional algorithms.

5) Discussion: Generally, 2K approximates the real network closest. Though, as we potentially want to utilize the topology synthesis for data augmentation, we do not require exact replicas of the input, which we anyways already have in the dataset. This is especially obvious for the DC, where the 2K-generated networks show zero variance. So, if we perform analyses where this centrality metric is an important KPI, we do not obtain additional variance in the data, as the DC directly relates to the JDM, which 2K is trying to duplicate.

GANs illustrate a viable alternative by inducing more variance as they do not rely on explicit information, while still being able to appropriately capture the original network metrics on average. Hence, it expresses variation in all of the chosen centrality metrics, including the DC, perfectly capturing the DC on average in most scenarios, even though it was only implicitly trained to do so. Though, the currently showcased GAN approach still leaves room for improvement. By feeding the GAN permutations of the real network, it is mainly able to maintain network characteristics, that persist throughout the node relabeling, e.g., star-like topologies or full meshs. When investigating topologies such as GtsCzechRepublic (as mentioned earlier), which consists of a more tree-, path-, or ring-like structure, the GAN may encounter inaccuracies, as the topology is optically not distinguishable from a randomly meshed network after permutating the node labels.

Lastly, the legacy generators perform well in selected scenarios, e.g., for more chaotic graphs ER is sufficient, and for star-like graphs BA performs well. Though, as each of those legacy algorithms has an underlying model it adheres to, none of them depict a general solution. Additionally, BA and WS show zero variance in the DC, as the number of links is fixed per design, while the average node degree of ER follows a Binomial distribution, thus showing more deviation.

VI. Synthesis of Weighted Topologies

After examining the synthesis of unweighted topologies, we focus on more use case-specific attributes in the following, namely the synthesis of weighted topologies, where the weight depicts the actual geographical distances of two nodes in a WAN, e.g., two switches/routers in different cities. For this, we examine the four networks as in the previous section, with respect to the weighted equivalents of the centrality metrics.

As the model-based generators generally showed inferior performance, we only focus on 2K- and GAN-generated networks for the remainder of this work. Note that both methods only generate unweighted graphs, so we generate a weighted graph by sampling the real weights onto the generated networks, which we call 2K and GAN_{BW}, respectively. However, for GAN, there is the additional possibility to generate the weights natively, i.e., via the previously shown color channels. Thus, we also investigate this much more implicit weighted network synthesis which does not expect any explicit information about the network, called GAN_{RGB}. Figure 6 depicts the average centrality metrics for the generated networks in comparison to the original four networks.

1) BREN: In Figure 6(a), we see the results for BREN as a box plot. For the BC, we see that GAN_{BW} performs best, as the GAN in the previous section achieved the best fitting results for the average BC, and simply sampling real distances has an advantage over the implicit approach of GAN_{RGB}. However, this also showcases a GAN's capability of just implicitly inferring the geographical distances, as GAN_{RGB} performs similar to 2K.

Furthermore, the results for the CC illustrate further advantages of the GAN_{RGB}. While all approaches are far off the real average (note the logarithmic y-axis), GAN_{RGB} performs better than both of the explicitly sampled approaches. As BREN consists of clusters with smaller distances which are connected via longer links, the wrong placement of these longer links is detrimental and influences the weighted graph metrics. Thus, GAN_{RGB} is at least capable of

placing these links not in middle of these clusters, while the other approaches do not make any differentiation when placing weights.

Lastly, as we sample geographical distances for 2K and GAN_{BW} directly from the real weights, it is expected that we achieve a perfect fit on average. While GAN_{RGB} performs not as perfect as GAN_{BW}, it still approximates the real network sufficiently by only implicitly inferring the weights.

2) BtNorthAmerica: Figure 6(b) illustrates the results for the weighted topology synthesis for BtNorthAmerica. Overall, 2K and GAN_{BW} perform very similar, with both GANs expressing slightly higher value ranges. Note that the variance of GAN_{BW} of the weighted DC is slightly higher than 2K, even though both approaches sample from the same weights, as we defined the weighted DC as the sum of all edge weights connected to a single node, which is directly influenced by the number of links, which in return has a greater variance for the GAN, as seen in the previous section. As depicted in the previous section, though, the metrics of BtNorthAmerica are more difficult to replicate by the algorithms, which becomes also apparent for the weighted metrics here. GAN_{RGB} performs worse for the CC as it predicts the edge weights implicitly, however, especially for the BC and DC, the performance approaches the two sampled versions.

3) GÉANT: Figure 6(c) depicts the results for GÉANT. We see comparable trends to BtNorthAmerica, with GAN_{RGB} performing slightly worse. However, the algorithms generally are better at approximating the real values. Additionally, we can observe a trade-off between 2K and GAN_{BW}, where 2K is able to match the BC better, and GAN_{BW} matches the CC better. Again, the GAN-based approaches also express slightly more variance.

4) GtsSlovakia: Lastly, Figure 6(d) illustrates the results for GtsSlovakia. We see an identical trend compared to GÉANT concerning the trade-off for the BC and the CC, and a similar trend concerning the value ranges of the generated networks.

5) Discussion: The analysis of the weighted topologies showcases, that 2K and GAN_{BW}, as expected, perform slightly better than GAN_{RGB}, as they explicitly infer information about the edge weights. However, it also illustrates that GAN_{RGB} is capable of implicitly inferring the weights appropriately, though it has a slight tendency to underestimate the distances compared to the real values, which directly correlates to overestimating the closeness. Although, in case of BREN even outperforms the other two approaches for the CC. Additionally, there seems to be a trade-off between 2K and GAN_{BW}, where 2K is generally better at approximating the BC, and GAN_{BW} is a better match for the CC, which can be seen when investigating GÉANT and GtsSlovakia. Lastly, we also observe that the correct placement of edge weights is important, as it can drastically change weighted metrics, which have great influence on the actual performance of a computer network, which we examine in the next section.

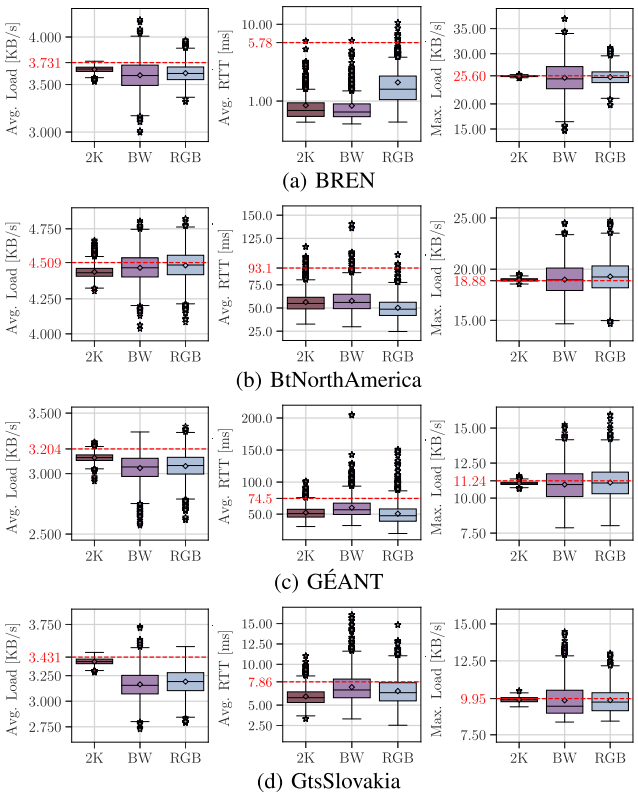


Fig. 7. Performance metrics for the four networks; red dashed line (---) depicts the metrics of the corresponding real network.

VII. Simulating the Networks

After investigating more theoretical graph metrics, this section aims at a more practical application of the synthesized networks by porting the topologies into the OMNeT++ simulator and running simulations in an SDN environment to analyze dynamic performance metrics. For 1,000 generated networks, three generators, four real-world WAN topologies, and four repetitions each this results in $4{\cdot}1{,}000{\cdot}3{\cdot}4 = 48{,}000$ simulation runs, equating to a real-world time of over 333 days. For both the RTT and controller load we investigate the average value. We also include the maximum controller load, but leave out the maximum RTT for sake of brevity. The results for average and maximum RTT showed similar trends, just on a different value scale.

1) BREN: Figure 7(a) shows the results for the simulated performance metrics for BREN as a boxplot. For the average load at the controller, all three approaches approximate the load appropriately, with 2K being slightly closer on average. The controller load directly correlates to the betweenness centrality, since a higher betweenness centrality of a node – an OpenFlow switch in our WAN simulation – means that a node is contained in many shortest paths. Thus, the refreshing rate of flow table entries is higher, increasing the hit ratio, and ultimately decreasing the amount of relayed traffic. As both the 2K- and GAN-generated networks depicted a similar betweenness to the real networks, this also shows here.

For the RTT, however, all approaches show great deviations from the actual performance, directly negatively correlated to

the weighted closeness metric, as the closer nodes to each other are, the lower the RTT is. Again, as GAN_{RGB} is at least capable of placing the longer links not directly into the highly meshed clusters, it performs better than the other two approaches (note the logarithmic y-axis again, similar to the analysis of the weighted CC).

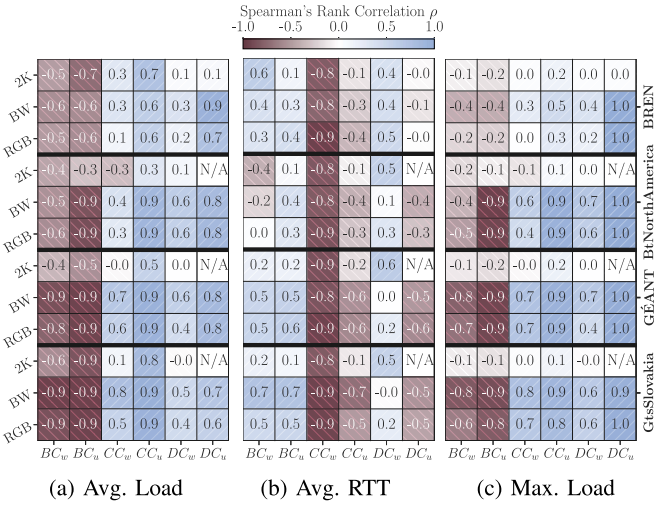
Lastly, the maximum controller load is on average perfectly approximated for all approaches. The maximum controller load is mostly dominated by the LLDP messages, which are sent out in regular intervals. As the switches are instructed to send out such a discovery message to every neighbouring switch, this directly correlates to the node degree. Noticeably, 2K even shows little variation for this performance metric, as the node degree in each generated network is the same.

2) BtNorthAmerica: The results for BtNorthAmerica are shown in Figure 7(b). Similar to before, all approaches approximate the average controller load appropriately. This time, the GAN-based approaches are slightly closer on average, however. While the RTT of the synthetic networks is generally closer than for BREN, it still suffers from similar problems. Lastly, the maximum load is on average again closely approximated by all approaches, with 2K showing little variance.

3) GÉANT: The results for GÉANT are shown in Figure 7(c). The three approaches all reasonably replicate the behavior of the average controller load, with 2K having a slight advantage. The RTT of the synthetic network is closest approximated by GAN_{BW}, and works generally better than for the other two networks. Thus, we can see the aforementioned trade-off also when investigating the performance metrics. The observations regarding the variance are seen here as well, especially visible for the maximum controller load, again.

4) GtsSlovakia: The results for GtsSlovakia are shown in Figure 7(d). We see an identical trend compared to GÉANT, i.e., we observe the trade-off between 2K and GAN-based approaches again, as well as the differences in variation for the approaches.

5) Discussion: The analysis of the performance metrics shows that an adequate replication of the graph metrics provides a robust basis for more practical use cases. The graph and performance metrics are highly correlated, i.e., the average load relates strongest to the BC, the average RTT to the CC, and the maximum load to the DC. Figure 8 investigates this further by depicting the correlation of the chosen performance and graph metrics for all three synthesis approaches of all four networks. Generally, both load-based metrics are more correlated to the unweighted graph metrics and also show a strong correlation to all three of them, whereas the RTT is mostly influenced by the weighted CC. Interestingly, the RTT shows only a weak to moderate correlation to the weighted DC, i.e., the sum of the adjacent edge weights. The figure also shows that the unweighted DC for the 2K-generated networks cannot be calculated for all networks besides BREN, which again emphasizes on the fact that the generated networks all have the same value for this centrality. Thus, when utilizing such networks, e.g., as an augmentation for ML input, it may undermine the importance of the metric.



(a) Avg. Load (b) Avg. RTT (c) Max. Load

Fig. 8. Correlations of performance metrics and weighted + unweighted graph metrics for all four networks.

While a close replication of graph metrics benefits the performance metrics, closer graph metrics do not automatically equate to more accurate performance metrics, e.g., in case of BREN the GAN-based approach approximated the BC and CC slightly better than 2K, but the load-based performance metrics where more accurate for 2K. The reverse is true for BtNorthAmerica.

Lastly, the GAN_{RGB}-generated networks are able to hold up with the 2K- and GAN_{BW}-generated networks by only inferring the edge weights implicitly. Though, we still see the aforementioned necessity of a correct placement of edge weights, especially visible for BREN.

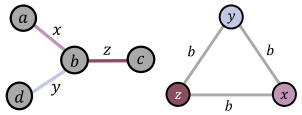
VIII. Sophisticated Attribute Sampling

In this section, we want to examine a more sophisticated way of sampling the weights onto the generated networks, instead of just drawing the weights uniformly. The rationale behind this is the fact that the weight of the edge is correlated to the actual unweighted topology, e.g., in our context a link with a long distance might connect two local areas, and as long links are expensive, there may be few redundant links, resulting in the link being contained in many paths from one local area to another. Similar ideas have been shown to be fruitful in the social network context [10], [54] with node attributes, utilizing node-based graph metrics, e.g., node degrees or joint degrees. We want to sample link attributes, and not many useful edge-based centrality metrics exist.

1) Edge-Based Graph Metrics: One of the few available metrics to compute for an edge is the edge betweenness centrality (EBC). Similar to the node BC, the (unweighted) EBC of an edge is the fraction of shortest paths, that the edge is contained in, and is computed as follows:

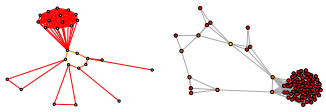
$$EBC_u(e(v_1, v_2)) = \sum_{v_3 \neq v_4} \frac{|\{sp_u(v_3, v_4) | e(v_1, v_2) \in sp_u(v_3, v_4)\}|}{|sp_u(v_3, v_4)|}$$

So, compared to the node BC, it also includes direct connections that traverse the edge. Note that the weighted EBC



(a) OG (b) LG

Fig. 9. Theory: OG → LG.



(a) OG (b) LG

Fig. 10. Praxis: OG → LG.

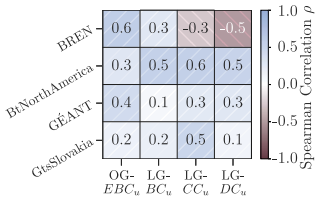


Fig. 11. Correlation of node distances/edge weights and the underlying topology for all four networks.

is computed analogously, but is not utilized here, as we are merely interested in the underlying layout to be able to sample the weights.

2) Line Graphs: Even though there might not be many edge-based graph metrics, we can still utilize node-based metrics by using line graphs (LGs). The concept behind LGs is illustrated in Figure 9, depicting a transformation from the edges of the original graph (OG) into the nodes of the LG. An edge between the nodes of the LG is created if there exists a node in the OG, that is connected to the two corresponding edges. A practical application in our context can be seen in Figure 10 with the example of BREN. When executing this transformation, we can also propagate the edge attributes of the OG to node attributes in the LG, e.g., the geographical distance is now a node attribute instead of an edge attribute, indicated by the colors in Figures 9 and 10. This allows us to calculate node-based centrality metrics, e.g., we opt to utilize the node-based centralities BC, CC, and DC, again, in addition to the EBC. To avoid confusion with the graph metrics we use as a quality measure, henceforth we will adhere to the sampling bases as OG-EBC, LG-BC, LG-CC, and LG-DC, which also reflects if the metric was calculated on the OG or the LG. Figure 11 depicts the correlation between the edge weights of the previously investigated networks and the four unweighted centralities, illustrating at least a moderate monotonic relationship.

3) Attribute Sampling: Calculating the four aforementioned centrality metrics now allows to perform attribute sampling with them as a basis, i.e., we construct a joint probability matrix $P(W, C)$ of edge weights W and centrality metric C from the real network. As the synthesized networks may express other centrality values than the original network and since the metrics are continuous, we apply binning to discretize the values, i.e., we perform a parameter study and bin the possible value range of the centralities into 1, 2, 4, 8, 16, 32, 64, and 128 bins. Smaller amounts of bins decrease the chance that a bin is empty, at the cost of accuracy, e.g., in case of a single bin we practically sample from a uniform distribution, again. To assign weights to the synthetic networks, we may now compute the respective centrality metric for each edge and can utilize the matrix to look up the conditional probabilities instead of a uniform distribution. For example, if an edge in the synthetic network has a centrality of c, we can utilize the conditional probability $P(W|C = c)$. This allows us to assign weights to edges that express similar structural characteristics compared to the real network. As we generally have sparse networks with few redundant links, the chance that a bin is empty is non-negligible. Thus, we set up a fallback mechanism, i.e., if all probabilities for a given centrality are 0, we look for the closest non-empty bin and utilize its probabilities.

4) Case Study: We study the approach of attribute sampling with BREN, as here the correct weight placement is the most crucial, and investigate weighted graph metrics and corresponding performance metrics.

Graph Metrics: Figure 12(a) illustrates the resulting weighted graph metrics of the sampling approach for all four investigated centralities as sampling basis for 2K and varying bin sizes. As the GAN_{RGB}-based approach already infers edge weights implicitly, the solid magenta line additionally depicts the performance of GAN_{RGB} from the previous section. As mentioned, the use of only a single bin essentially equates to the random sampling to grant a baseline.

For the weighted BC, we generally see a decrease in accuracy for all four sampling bases when more bins are utilized, with the LG-DC-based sampling decreasing the least. In contrast, for the weighted CC a vast improvement can be seen for all sampling bases. Though, OG-EBC and LG-DC seem to be more stable with respect to outliers, as for LG-BC and especially LG-CC the mean and median values are diverging drastically. Lastly, for the weighted DC, all sampling bases besides LG-CC are fitting with LG-DC being on point for all bin amounts. Consequently, LG-DC seems to be an appropriate attribute sampling base for the 2K-based approach, which is inline with the idea of the 2K algorithm, as it is based on the (joint) node degrees anyways.

Analogously, Figure 12(b) illustrates the weighted graph metrics for the GAN_{BW}-based approach. Similarly, the accuracy for the weighted BC also decreases here for all sampling bases, though, the decrease is not as severe. For the weighted CC the trend is also similar to the 2K approach, as for all bases the metrics get closer. However, only LG-DC provides a robust base. Lastly, the weighted DC, however, diverges drastically for LG-DC and LG-CC as base the more bins are used. This may be a result of the fact, that the GAN obfuscates BREN more, i.e., while 2K keeps the big cluster of nodes isolated, the GAN may add extra edges here. To balance these extra edges out, they also need to have similar weights like the edge that initially connects the cluster to the rest of the network.

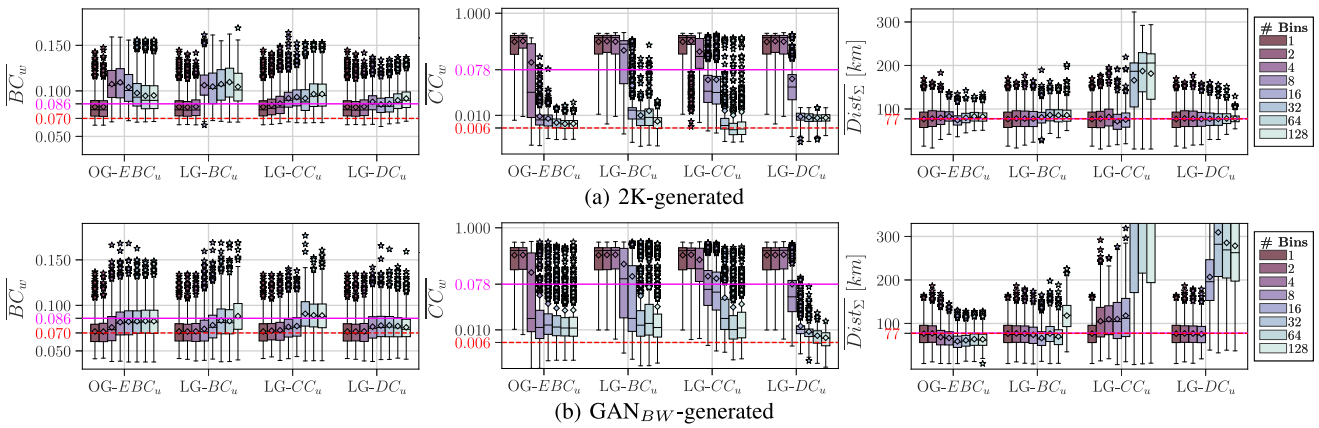


Fig. 12. Comparison of weight attribute sampling methods for generated BREN-like topologies; red dashed line (---) depicts the metrics of BREN itself, magenta solid line (—) depicts the metrics (avg.) of the GAN$_{RGB}$-generated networks.

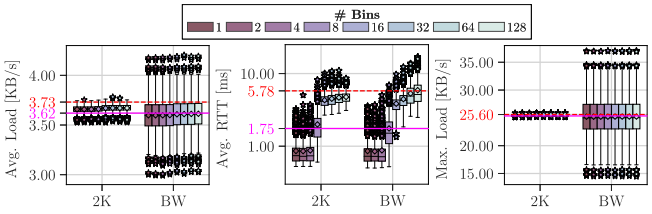


Fig. 13. Performance metrics with LG-DC as sampling base; red dashed line (---) depicts BREN itself, magenta solid line (—) depicts GAN$_{RGB}$-generated networks.

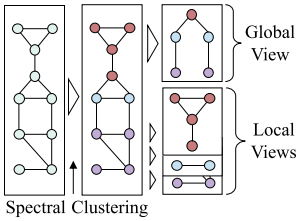


Fig. 14. Exemplary visualization of global and local views via spectral clustering for hierarchical synthesis.

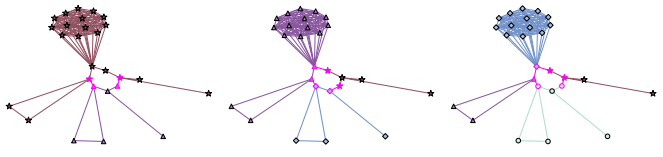
Performance Metrics: To investigate the effect of the attribute sampling on the performance metrics, Figure 13 illustrates the results, adding another $1{,}000 \cdot 2 \cdot 8 \cdot 4 = 64{,}000$ simulation runs, for two generators, eight bin numbers and four repetitions for all 1,000 generated topologies, equating to a real-world time of over 444 days. For brevity's sake, we limit ourselves to the results with LG-DC as sampling base.

For the average controller load, the influence of the number of bins is only marginal. In this scenario, the controller load is mainly influenced by the unweighted BC. As mentioned earlier, the BC directly relates to the flow table hit ratio, which is a major part of the traffic relayed to the controller. As the utilized routing algorithm here is based on the shortest paths with respect to the amount of hops, the distance between the nodes is not as important. The main difference between 2K and GAN is the value range. For the RTT, we see a major improvement, in direct relation to the weighted CC. While GAN performs more accurate than 2K here, this may look different for a different sampling base, e.g., OG-EBC is also a suitable option for 2K. Interestingly, even though the GAN samples in sum higher distances onto the link compared to the original approach, this actually helps balancing out the RTT, as no shortcut via a short link into the cluster can be taken now, as explained earlier. Lastly, the maximum controller load is not influenced by binning, as it is mainly related to the number of links, not their weight/distance, and was perfectly replicated in the original approach anyway.

Discussion: The analysis of sophisticated attribute sampling shows that it can depict an effective way of combating the problem we have encountered with the weighted CC. Though, it may come at the cost of a slight degradation of other graph metrics, as seen with the weighted BC and partly with the DC. In our case we did not see a negative influence on the actual performance metrics. However, this may vary dependent on the context/setup, e.g., using a latency-based routing algorithm instead of a hop-based one. The approach works well if there is some sort of relationship between edge weights and edge centralities. This relation does not necessarily have to be monotonic, i.e., show a correlation, but can map the weights and centralities in other ways (e.g., odd/even degrees $\hat{=}$ high/low weights). For BREN, OG-EBC and LG-DC show the highest correlation to the edge weights, both are most stable for 2K, and LG-DC for GAN. Lastly, while the attribute sampling may be a feasible way to generate weights in a postprocessing step for 2K and GAN$_{BW}$, it defeats the purpose of using the implicit approach of GAN$_{RGB}$.

IX. Hierarchical Topology Synthesis

In this section, we want to test another approach that also may help GAN_{RGB} improving edge weight assignments, thus, in a more implicit manner. Here, we start by dividing the topology into different clusters, and then proceed with the same methodology as in the original approach for the resulting clusters. Though, we lower the amount of permutation matrices fed into the GAN to 5,000, as the clusters are much smaller than the whole topology. The core idea is depicted in Figure 14. We distinguish between local and global parts of the network,



(a) Two clusters. (b) Three clusters. (c) Four clusters.

Fig. 15. BREN after applying spectral clustering; magenta outline (—) depicts nodes/edges contained in the global view, other colors/shapes represent local views.

thus, by augmenting these parts separately, we shift the weight assignment towards the desired direction.

1) Spectral Clustering: Spectral clustering is an unsupervised ML-algorithm highly suitable for graphs, which takes as input a similarity matrix to cluster a graph. This type of clustering is especially useful here, as we only have our distances as features, so it is basically a 1D clustering, rendering traditional methods such as *k*-means inapplicable.

The similarity matrix can be computed via the given physical distances of nodes. By computing the weighted shortest paths between all node pairs, we interpret these distances as a measure of dissimilarity. We convert the corresponding dissimilarity matrix D into a similarity matrix S, by first normalizing D via min/max-normalization, and then define $S = 1 - D_{norm}$. We cluster S with sklearn's implementation, and set the number i of clusters to 2, 3, and 4, and call the clusters C_j *local views*, with edges E_{local_j} and nodes V_{local_j} with cluster IDs $j \in [0, i)$:

$$V_{local_j} = \{v \in C_j\}$$

$$E_{local_j} = \{e(v_1, v_2) \in E | v_1 \in C_j \cap v_2 \in C_j\}$$

The local views are connected via the residual edges not contained in the clusters, which we call the *global view*, with edges E_{global} and nodes V_{global}:

$$V_{global} = \{v_1 \in C_j | \exists v_2 \in N(v_1) \cap v_2 \notin C_j\} \; \forall j$$

$$E_{global} = \{e(v_1, v_2) \in E | v_1 \in C_j \cap v_2 \in C_k \cap j \neq k\} \; \forall j, k$$

2) Case Study: Again, for our evaluation of the hierarchical graph synthesis we focus on BREN, as here the challenge of placing the edge weights correctly was the most difficult. The resulting clusters are illustrated in Figure 15.

Graph Metrics: Figure 16(a) illustrates the results for the graph metrics of the hierarchical synthesis for the naïve approach with one cluster, as well as three different amounts of local views. For the weighted BC, we observe that the approximation first slightly improves for more clusters, but then worsens again for both GAN-based approaches, and introduces no further benefit for 2K. As we create several clusters, we also introduce overhead as we have to merge the partial views again, potentially influencing the graph metrics negatively, when we make the clusters too fine-grained.

For the weighted CC, on the contrary, we see a drastic improvement for the approximation for all approaches. As

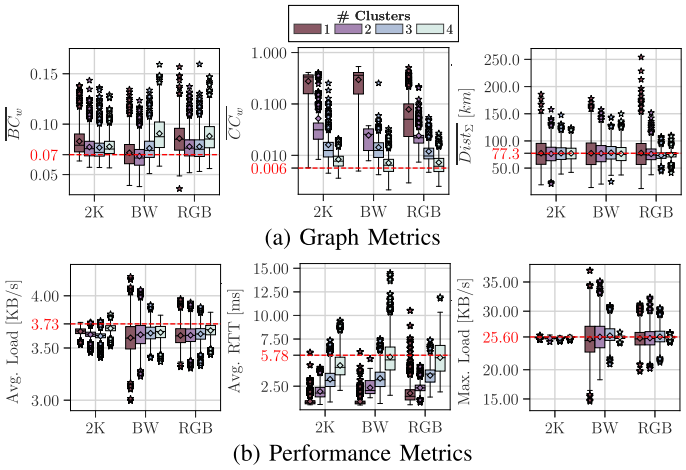


Fig. 16. Hierarchical synthesis of BREN; red dashed line (---) depicts the metrics of BREN itself.

we cluster the network into local views and a global view, we more or less force the generators to place the longer distances at plausible locations. Both GAN-based approaches are able to match the CC the most accurate for four clusters here.

For the DC, we see that the clustering reduces the value range, as we narrow down the problem and thus remove some of the inference work with this preprocessing. Though, the GAN still has a slight tendency to underestimate the distances as seen before. Naturally, the smaller the views are, the more probable it is that the GAN will produce a perfect (partial) fit, as the search space is limited, the variety in possible permutation matrices is low, and thus it will produce a valid sample more likely. In conjunction with this lack of input diversity, it is observable that small views are more prone to mode collapse, i.e., produce potentially convincing/high quality samples of limited output variety [55], another reason for reduced variance.

Performance Metrics: Figure 16(b) illustrates the results for the performance metrics, adding another 1,000 · 3 · 4 · 3 = 36,000 simulation runs, for three generators, three additional cluster sizes and four repetitions for all 1,000 generated topologies, equating to 250 days.

For the average controller load, we see a slight improvement in accuracy compared to the original approach with one cluster, especially for the GAN-based approaches. This is interesting, as the BC generally worsened, and in previous analyses this usually directly resulted in the controller load also being less accurate. Thus, we also investigated the unweighted BC, but it shows a similar trend to its weighted counterpart. The replication of the RTT improves drastically the more clusters are utilized, directly related to the CC. Lastly, the maximum controller load mainly changes its value range, especially for four local views and both GAN-approaches, as in the four cluster approach the big cluster of BREN is perfectly isolated now. As observed many times, the 2K approach shows little variance here, independent from the number of clusters.

TABLE II
PROS AND CONS OF THE VARIOUS GRAPH GENERATION APPROACHES

	2K	GAN	BA, ER & WS
Pro	+ Close approximation of all networks/metrics + Fast computation, no training needed	+ Flexible, i. e., no fixation on specific metrics + Native support of weights/distances + Close approximation for most networks/metrics	+ Very simple and fast algorithms + No explicit network information needed + Work well if the underlying model fits
Con	− Fixation of metrics, e. g., joint degrees − No support for weight synthesis	− Struggles with permutation dependency − Training needed (> 1 hour) − Complex post-/preprocessing and configuration	− Optimal parameters varying − No support for weight synthesis − Possible fixation of metrics, e. g., links

Discussion: The analysis of the hierarchical synthesis shows that it is also an effective way of tackling the synthesis of the weighted CC for BREN. Similar to the sophisticated attribute sampling though, this might also come at the cost of a slightly decreased accuracy for the BC, but this time, the DC can be preserved for all approaches. Also similar to the sophisticated attribute sampling, the effect of this trade-off on the performance metrics is mostly marginal, even if some of the graph metrics diverge. This approach works best if there are distinguishable clusters in the network, e.g., somewhat isolated, possibly more inter-meshed parts of the networks with low distances, that are connected to other parts via few redundant links. Lastly, this approach is also fitting for GAN_{RGB}, thus performing a more implicit inference.

X. DISCUSSION

As we have performed several analyses, we summarize the main pro and cons for each algorithm observed in those analyses in Table II. 2K is generally very close in the approximation of the real networks, and compared to GANs a fast algorithmic approach, that is still being extended. Though, it takes as input explicit information about the network, which it tries to replicate perfectly, resulting in no variation for metrics such as the DC. This possibly renders it less suitable for data augmentation tasks, where the DC is a parameter we want to investigate. Furthermore, there is no support for weighted graphs or other edge attributes.

GANs depict a DL-based approach and are highly flexible, as they only infer the information implicitly and thus also show variation for, e.g., the DC, while still being able to match it appropriately. GANs also support the weight synthesis natively by utilizing the color channels, and possible allow for even more attributes, such as bandwidth, to be encoded into the input. However, as we fed the GAN permutations of the real matrix, some information, e.g., long paths or ring structures may be lost. GANs are complex, thus need a training time, more processing, and are sensitive to configure.

The legacy algorithms are very simple approaches which do not infer any information explicitly. We also observe that they perform decently, if the underlying model is fitting. However, the algorithms have one or more parameters that need to be configured, and sometimes cannot be configured appropriately at all. They also support no weight synthesis and similar to 2K, for some algorithms we also have a fixation on metrics, e.g., for BA and WS we fixate the number of edges.

TABLE III
PROS AND CONS FOR IMPROVING WEIGHT ASSIGNMENTS

	Attribute Sampling	Hierarchical Synthesis
Pro	+ Fast and simple + Flexible sampling bases	+ Implicit inference possible + (Potentially) controllable variance
Con	− Only explicit inference − Struggles with sparsity	− Varying parameter settings − Merging overhead

Furthermore, we have also investigated two potential approaches to enhance the assignment of edge weights for weighted graphs, both showing promising results and summarized in Table III. Sophisticated attribute sampling is a fast and simple approach to assign the edge weights in a more sensible manner than just uniformly sampling by utilizing joint probabilities. Many metrics can be chosen as sampling bases and easily switched out. Though, this approach generally samples weights from the real network, thus, does not infer them implicitly. The sampling may potentially be problematic, as WANs are generally sparser networks, possibly resulting in many zero entries in the joint probability matrix.

A more implicit approach is the hierarchical synthesis. By dividing the network into local and global views, we enforce the weight assignment this way. By choosing the granularity via the number of clusters, we can potentially control the variance to a certain degree. Though, due to the clustering we may end up with clusters of greatly varying sizes that all possibly need different parameter settings for the GAN-based approaches. Lastly, as we divide the generation task into several subtasks, we need to puzzle these parts together again, resulting in merging overhead and room for inaccuracies.

XI. CONCLUSION

The synthetic generation of graphs dates back decades and has been a focus of researchers in various fields of application since then. Related work presents us with model-dependent, model-independent, and Deep Learning (DL)-based graph generators, which all have their respective advantages. As public datasets are limited, this paper compared different methods for synthetic graph generation in the context of data augmentation for Wide Area Networks (WANs) research, by comparing the synthetic networks on the basis of theoretical graph metrics, as well as more practical performance metrics in the context of software-defined networks. For the latter, we simulated almost three years of real-world time to conduct our analyses. The results illustrate that while algorithmic graph

generators approximate the original networks closely, by fixating explicitly on a metric, they potentially do not introduce enough desired variance for some of the metrics. Generative Adversarial Networks (GANs), on the other hand, do not infer any information explicitly, while still being able to appropriately match the real networks. However, there is still room for improvement, as GANs were not able to capture all characteristics of the chosen networks. In addition to a naïve graph generation, we also explored more sophisticated measures via attribute sampling and hierarchical synthesis to genete graph weights more appropriately.

In the future, we aim to broaden the scale of our evaluations by augmenting a multitude of networks instead of in-depth analyses of single networks. This allows us to apply Machine Learning (ML)-based tasks, e. g, performance prediction, and enables investigations of which generation methods yield a benefit for enriching a data set and/or what amount of variance is desired/needed. Additionally, we aim to enhance the GAN approach by employing punishments/rewards to the loss function or the generator/discriminator itself, e.g., punishing the generator if it creates disconnected graphs or other undesired properties, already been shown to be fruitful in other research areas [27]. We may also extend our research to more DL-based graph generation methods, such as Variational Autoencoders (VAEs) or Reinforcement Learning (RL)-based strategies.

REFERENCES

[1] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

[2] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0—Survivable network design library," *Netw. Int. J.*, vol. 55, no. 3, pp. 276–286, 2010.

[3] K. Sawada, D. Kotani, and Y. Okabe, "Network routing optimization based on machine learning using graph networks robust against topology change," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, 2020, pp. 608–615.

[4] K. Dietz, N. Gray, M. Seufert, and T. Hossfeld, "ML-based performance prediction of SDN using simulated data from real and synthetic networks," in *Proc. IEEE/IFIP Netw. Operat. Manage. Symp. (NOMS)*, 2022, pp. 1–7.

[5] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[6] P. Erdös and A. Rényi, "On the evolution of random graphs," in *The Structure and Dynamics of Networks*. Princeton, NJ, USA: Princeton Univ. Press, 2011, pp. 38–82.

[7] D. J. Watts and S. H. Strogatz, "Collective dynamics of "small-world" networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[8] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[9] T. Hou, T. Wang, Z. Lu, and Y. Liu, "Combating adversarial network topology inference by proactive topology obfuscation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 6, pp. 2779–2792, Dec. 2021.

[10] B. Tillman, A. Markopoulou, M. Gjoka, and C. T. Butt, "2k+ graph construction framework: Targeting joint degree matrix and beyond," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 591–606, Apr. 2019.

[11] I. Goodfellow et al., "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[12] "Base code." Accessed: Jul. 5, 2023. [Online]. Available: https://github.com/lsinfo3/cnsm2022-wan-topology-synthesis

[13] "Extended code." Accessed: Jul. 5, 2023. [Online]. Available: https://github.com/lsinfo3/tnsm2023-wan-topology-synthesis

[14] K. Dietz, M. Seufert, and T. Hoßfeld, "Comparing traditional and GAN-based approaches for the synthesis of wide area network topologies," in *Proc. 18th Int. Conf. Netw. Service Manage. (CNSM)*, Thessaloniki, Greece, Oct. 2022, pp. 64–72.

[15] M. E. Newman and J. Park, "Why social networks are different from other types of networks," *Phys. Rev. E*, vol. 68, no. 3, 2003, Art. no. 36122.

[16] F. Faez, Y. Ommi, M. S. Baghshah, and H. R. Rabiee, "Deep graph generators: A survey," *IEEE Access*, vol. 9, pp. 106675–106702, 2021.

[17] J. M. Hernández and P. Van Mieghem, *Classification of Graph Metrics*. Mekelweg, The Netherlands: Delft Univ. Technol., 2011, pp. 1–20.

[18] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "NetGAN: Generating graphs via random walks," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 610–619.

[19] M. He, M.-Y. Huang, and W. Kellerer, "Optimizing the flexibility of SDN control plane," in *Proc. IEEE/IFIP Netw. Operat. Manage. Symp.*, 2020, pp. 1–9.

[20] E. Kaljic, A. Maric, P. Njemcevic, and M. Hadzialic, "A survey on data plane flexibility and programmability in software-defined networking," *IEEE Access*, vol. 7, pp. 47804–47840, 2019.

[21] R. Barrett, A. Facey, W. Nxumalo, J. Rogers, P. Vatcher, andM. St-Hilaire, "Dynamic traffic diversion in SDN: Testbed vs Mininet," in *Proc. Int. Conf. Comput. Netw. Commun. (ICNC)*, 2017, pp. 167–171.

[22] J. Alcorn, S. Melton, and C. E. Chow, "SDN on-the-go (OTG) physical testbed," in *Proc. IEEE Conf. Depend. Secure Comput.*, 2017, pp. 202–208.

[23] N. Gray, T. Zinner, S. Gebert, and P. Tran-Gia, "Simulation framework for distributed SDN-controller architectures in OMNeT++," in *Proc. Int. Conf. Mobile Netw. Manage.*, 2016, pp. 3–18.

[24] S. S. Bhavanasi, L. Pappone, and F. Esposito, "Dealing with changes: Resilient routing via graph neural networks and multi-agent deep reinforcement learning," *IEEE Trans. Netw. Service Manag.*, early access, Jun. 20, 2023, doi: 10.1109/TNSM.2023.3287936.

[25] Q. Wu, R. Gao, and H. Zha, "Bridging explicit and implicit deep generative models via neural stein estimators," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 11274–11286.

[26] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*.

[27] N. De Cao and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," 2018, *arXiv:1805.11973*.

[28] M. Simonovsky and N. Komodakis, "GraphVAE: Towards generation of small graphs using variational autoencoders," in *Proc. Artif. Neural Netw. Mach. Learn. (ICANN)*, 2018, pp. 412–422.

[29] H. Wang et al., "Learning graph representation with generative adversarial nets," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 8, pp. 3090–3103, Aug. 2021.

[30] L. Zhang, L. Zhao, S. Qin, D. Pfoser, and C. Ling, "TG-GAN: Continuous-time temporal graph deep generative models with time-validity constraints," in *Proc. Web Conf.*, 2021, pp. 2104–2116.

[31] X. Guo, L. Wu, and L. Zhao, "Deep graph translation," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 17, 2022, doi: 10.1109/TNNLS.2022.3144670.

[32] S. Tavakoli, A. Hajibagheri, and G. Sukthankar, "Learning social graph topologies using generative adversarial neural networks," in *Proc. Int. Conf. Social Comput. Behav.-Cult. Model. Predict.*, 2017. [Online]. Available: http://ial.eecs.ucf.edu/pdf/Sukthankar-SBP2017LateBreak.pdf

[33] W. Liu, P.-Y. Chen, F. Yu, T. Suzumura, and G. Hu, "Learning graph topological features via GAN," *IEEE Access*, vol. 7, pp. 21834–21843, 2019.

[34] N. Gray, K. Dietz, and T. Hossfeld, "Simulative evaluation of KPIs in SDN for topology classification and performance prediction models," in *Proc. 16th Int. Conf. Netw. Service Manage. (CNSM)*, 2020, pp. 1–9.

[35] A. Grigorjew, M. Seufert, N. Wehner, J. Hofmann, and T. Hoßfeld, "ML-assisted latency assignments in time-sensitive networking," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, 2021, pp. 116–124.

[36] M. J. Alenazi and J. P. Sterbenz, "Comprehensive comparison and accuracy of graph metrics in predicting network resilience," in *Proc. 11th Int. Conf. Design Reliable Commun. Netw. (DRCN)*, 2015, pp. 157–164.

[37] K. Yamashita and H. Ohsaki, "On the correlation between spectral measures of network topology and flow-level QoS," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, 2021, pp. 425–428.

[38] "OOS NED files." Accessed: Jul. 5, 2023. [Online]. Available: https://github.com/lsinfo3/OpenFlowOMNeTSuite/tree/master/openflow/networks

[39] "Deep convolutional generative adversarial network." Accessed: Jul. 5, 2023. [Online]. Available: https://github.com/tensorflow/docs/blob/master/site/en/tutorials/generative/dcgan.ipynb

[40] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*.

[41] A. Hagberg, P. Swart, and D. S. Chult, "Exploring network structure, dynamics, and function using NetworkX," Los Alamos Nat. Lab. (LANL), Los Alamos, NM, USA, Rep. AC52-06NA25396, 2008.
[42] V. Zadorozhnyi and E. Yudin, "Growing network: Nonlinear extension of the Barabasi-albert model," in *Proc. Int. Conf. Inf. Technol. Math. Model.*, 2014, pp. 432–439.
[43] K. Lei, M. Qin, B. Bai, G. Zhang, and M. Yang, "GCN-GAN: A non-linear temporal link prediction model for weighted dynamic networks," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 388–396.
[44] F. Loh, N. Mehling, S. Geissler, and T. Hoßfeld, "Graph-based gateway placement for better performance in LoRaWAN deployments," in *Proc. 20th Mediterr. Commun. Comput. Netw. Conf. (MedComNet)*, 2022, pp. 190–199.
[45] M. J. Alenazi and E. K. Cetinkaya, "Resilient placement of SDN controllers exploiting disjoint paths," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 2, 2020, Art. no. e3725.
[46] S. Lange, A. Grigorjew, T. Zinner, P. Tran-Gia, and M. Jarschel, "A multi-objective heuristic for the optimization of virtual network function chain placement," in *Proc. 29th Int. Teletraffic Congr. (ITC)*, vol. 1, 2017, pp. 152–160.
[47] M. K. Forland, K. Kralevska, M. Garau, and D. Gligoroski, "Preventing DDoS with SDN in 5G," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2019, pp. 1–7.
[48] N. McKeown et al., "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.
[49] "OOS exemplary config file." Accessed: Jul. 5, 2023. [Online]. Available: https://github.com/lsinfo3/OpenFlowOMNeTSuite/blob/master/openflow/scenarios/fattree/Scenario_DynamicFatTree_ARP_Ping_Drop.ini
[50] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proc. 2nd USENIX Workshop Hot Topics Manage. Internet Cloud Enterprise Netw. Services (Hot-ICE)*, 2012. [Online]. Available: https://www.usenix.org/system/files/conference/hot-ice12/hotice12-final33_0.pdf
[51] N. Gude et al., "NOX: Towards an operating system for networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, 2008.
[52] "INET framework." Accessed: Jul. 5, 2023. [Online]. Available: https://inet.omnetpp.org/
[53] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," 2016, *arXiv:1701.00160*.
[54] M. Seufert, S. Lange, and T. Hoßfeld, "More than topology: Joint topology and attribute sampling and generation of social network graphs," *Comput. Commun.*, vol. 73, pp. 176–187, Jan. 2016.
[55] S. Adiga, M. A. Attia, W.-T. Chang, and R. Tandon, "On the tradeoff between mode collapse and sample quality in generative adversarial networks," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, 2018, pp. 1184–1188.

Katharina Dietz (Student Member, IEEE) received the master's degree in computer science from the University of Würzburg, Germany, in 2020, where she is currently pursuing the Ph.D. degree. She is a Research Assistant with the Chair of Communication Networks, University of Würzburg. Her research mainly focuses on managing and securing communication networks with machine learning-based approaches, ranging from performance prediction to anomaly detection.

Michael Seufert (Member, IEEE) received the bachelor's degree in economathematics and the Diploma, Ph.D., and Habilitation degrees in computer science from the University of Würzburg, and holds the First State Examination degree in mathematics, computer science, and education for teaching in secondary schools. He has been a Research Group Leader with the Chair of Communication Networks, University of Würzburg, Germany, since 2019. His research focuses on quality of experience of Internet applications, network management, artificial intelligence, and machine learning for communication networks, as well as performance analysis and modeling of communication systems.

Tobias Hoßfeld (Senior Member, IEEE) has been a Full Professor and the Head of the Chair of Communication Networks, University of Würzburg, Germany, since 2018. From 2014 to 2018, he was the Head of the Chair "Modeling of Adaptive Systems" with the University of Duisburg–Essen, Germany. Among others, he received the Fred W. Ellersick Prize 2013 (IEEE Communications Society). He is a member of the editorial board of IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, ACM SIGMM Records, and *Quality and User Experience* (Springer).