Comparing Traditional and GAN-based Approaches for the Synthesis of Wide Area Network Topologies

Katharina Dietz, Michael Seufert, Tobias Hoßfeld
University of Würzburg, Germany
{katharina.dietz, michael.seufert, tobias.hossfeld}@uni-wuerzburg.de

***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 applications 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. Moreover, we investigate a hierarchical graph synthesis approach, which divides the synthesis into local clusters. 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.**

***Index Terms*—Wide Area Networks (WAN), Generative Adversarial Networks (GAN), Graph Generation, Graph Synthesis, Network Topology, Spectral Clustering.**

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. In the latter case, 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 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 (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.
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.

978-3-903176-51-5

3) We investigate a hierarchical graph synthesis approach, which divides the synthesis into local clusters.
4) We publish our code and generated networks [12].

The remainder of this work is structured as follows. Firstly, in Section II, we give background information about network topologies, algorithmic graph generators, and GANs, while Section III outlines related work. In Section IV, we specify the utilized networks, graph generators, and graph metrics. We then evaluate the generators' capability to synthesize unweighted network topologies similar to the real networks in Section V. In Section VI we adapt the initial approach to generate weighted graphs, representing WANs, which we further extend in Section VII, where we study a hierarchical graph synthesis approach. Section VIII provides a discussion by highlighting the respective (dis)advantages of the studied approaches. Lastly, we summarize our findings and provide an outlook for future work in Section IX.

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 metrics, which may be used for graph 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 [13]. 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.

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 $v \in V$
$sp_u(v_1, v_2)$	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.

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 [14], 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. [15], 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.

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 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.

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.

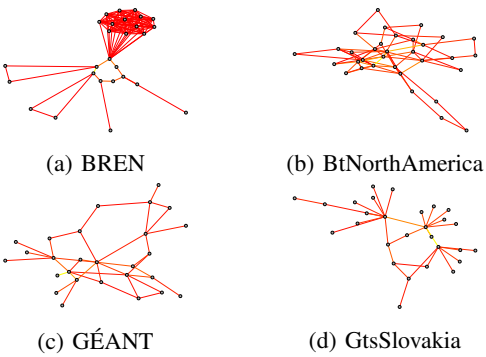
The above approaches are mainly geared towards social networks and have been tested thoroughly in this field of application. The goal of this work is to investigate these approaches in the context of WANs by providing an extensive analysis of weighted and unweighted graph metrics.

Graph Generation via GANs. An alternative to algorithmic approaches is DL-based graph generation. As showcased by Faez et al. [16], research for such deep graph generators mainly focuses on chemical and bioinformatics [17], social structures [15], [18]–[20], or synthetic datasets [19], [20], 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. [21] for the general approach, and Liu et al. [22] 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 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.

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 conclude this section by defining the graph metrics we use for comparing synthetic networks to the real network.



(a) BREN (b) BtNorthAmerica

(c) GÉANT (d) GtsSlovakia

Fig. 1: The chosen WANs from the ITZ – red links depict small geographical distances (in relation to the respective maximum distance), yellow links bigger ones.

A. Chosen Networks

We pick four networks from the ITZ, which are depicted in Figure 1, as they cover topologies with varying characteristics. In detail, we utilize a subset of the ITZ provided by Gray et al. [23], [24] 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.

a) 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.

b) 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.

c) 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.

d) 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 [25], which we use for the algorithmic generators.

a) 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.

b) 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 $2 \cdot m_{BA}$ [26, 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: $m_{BA} = \left\lfloor \frac{m}{n} \right\rceil$. 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.

c) 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: $k_{WS} = \left\lfloor \frac{2m}{n} \right\rceil$.

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.

d) 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.

e) 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 2. The top row illustrates the four networks from Figure 1 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 1, 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 [27]. 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.

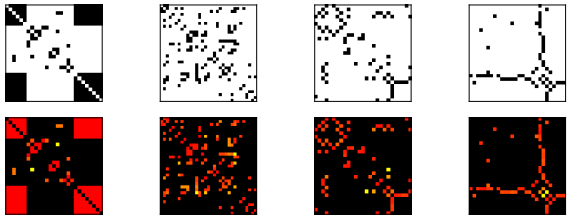


Fig. 2: BREN, BtNorthAmerica, GÉANT, and GtsSlovakia as images (left to right), top: as BW, bottom: as RGB.

As proposed by Tavakoli et al. [21], 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 3 illustrates the architecture of our utilized GAN. The architecture is adapted from [28], 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 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

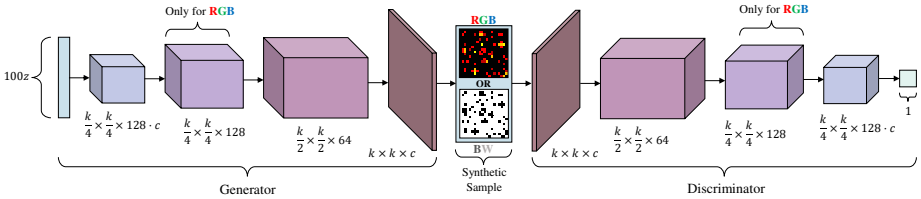


Fig. 3: Architecture of the utilized GAN.

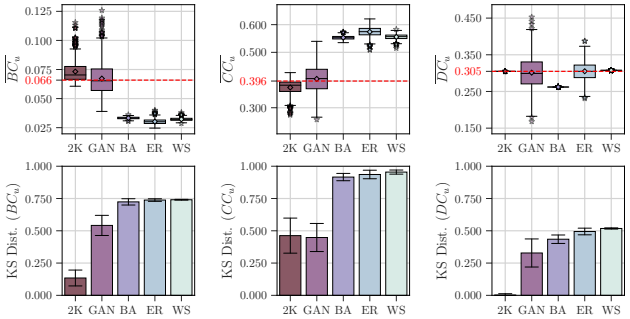


Fig. 4: Unweighted synthesis for BREN.

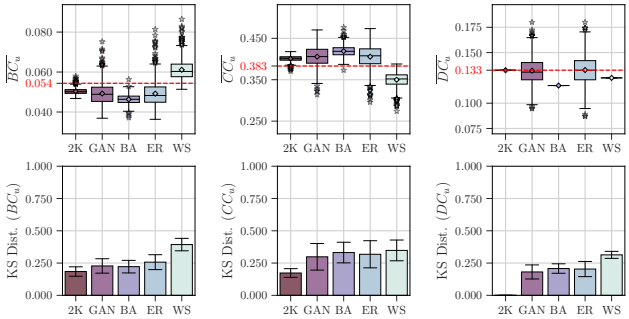


Fig. 5: Unweighted synthesis for BtNorthAmerica.

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) [29], SDN-enabled performance prediction [4], LoRaWAN gateway placement [30], SDN controller placement [31], or Virtualized Network Function (VNF) placement [32]. 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:

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

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 [33], we generate these 1k networks from ten different seeds to obtain robust results.

1) BREN: Figure 4 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

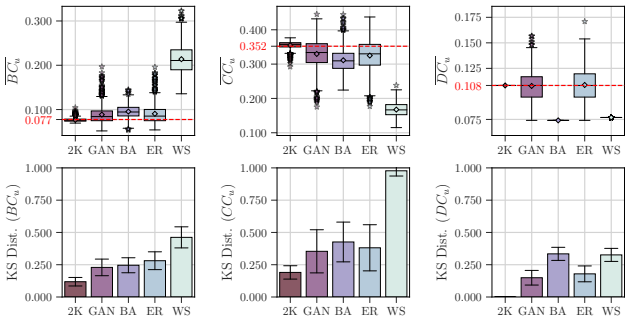


Fig. 6: Unweighted synthesis for GÉANT.

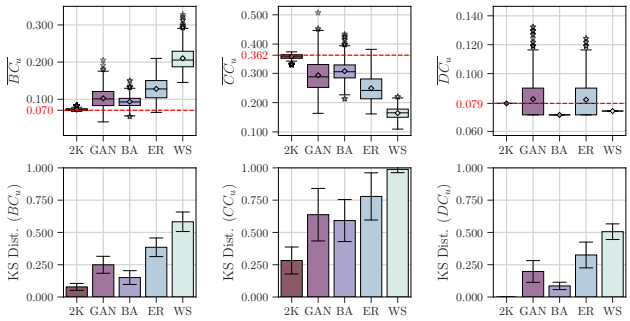


Fig. 7: Unweighted synthesis for GtsSlovakia.

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 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 6 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.

4) GtsSlovakia: Figure 7 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.

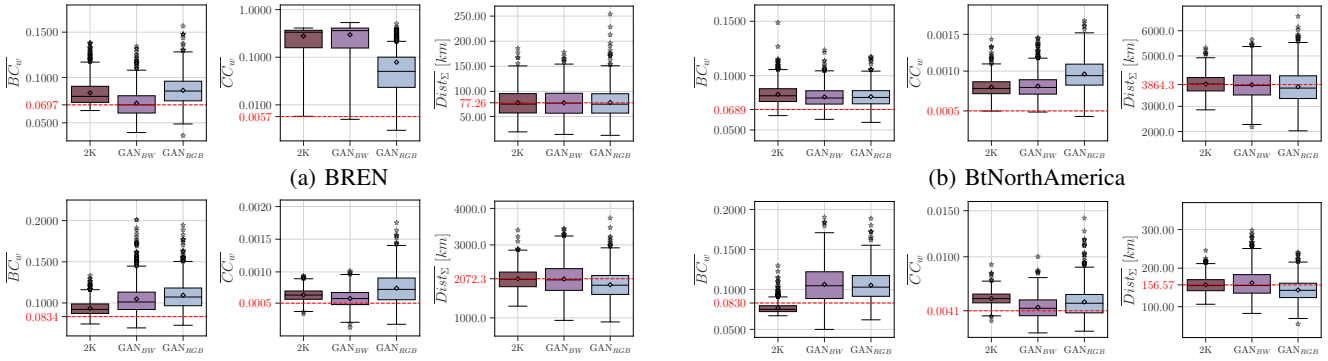
GANs illustrate a viable alternative by inducing more variance as they do not rely on explicit information, while still being able to appropriately capture the original network metrics on average. Hence, it expresses variation in all of the chosen centrality metrics, including the DC, perfectly capturing 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 net-



(a) BREN (b) BtNorthAmerica

(c) GÉANT (d) GtsSlovakia

Fig. 8: Weighted synthesis for the four networks.

works, 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 8 depicts the average centrality metrics for the generated networks in comparison to the original four networks.

1) BREN: In Figure 8a, 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 8b 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 8c 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 8d illustrates the results for GtsSlovakia. We see an identical trend compared to BtNorthAmerica 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 we examine in the next section.

VII. Hierarchical Topology Synthesis

In this section, we investigate a possible solution for the correct placement of edge weights, focusing especially on long-distance links, which have a great impact on WANs and their performance. We start by dividing the topology into different clusters, and then proceed with the same methodology as before for the resulting clusters. Though, we lower the

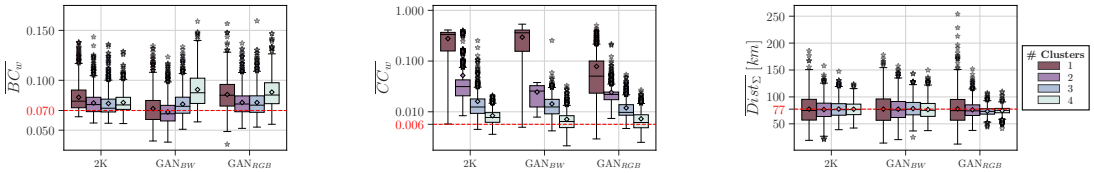


Fig. 9: Hierarchical synthesis of BREN.

amount of permutation matrices fed into the GAN to 5,000, as the clusters are much smaller than the whole topology.

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: 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. Figure 9 illustrates the results for 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 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, especially visible for 2K and GAN_{RGB}, 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 [34], another reason for reduced variance.

VIII. 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 to synthesize networks. They 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 on average. 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 also complex, thus need a training time, more processing, and are also sensitive to configure.

Lastly, 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.

IX. 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 DL-based graph generators, which

TABLE II: Summary of pros and cons of the various graph generation approaches.

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

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. 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 graph 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.

Future work can pursue two main directions. Firstly, we aim to improve the current approaches by employing more sophisticated measures of adding weights to the generated graphs, e. g., by employing attribute sampling similar to Seufert et al. [35], and also enhance the GAN approach by employing punishments/rewards to the loss function, e. g., punishing the generator if it creates disconnected graphs or other undesired properties. Similar approaches have already been shown to be fruitful in other research areas [17].

Secondly, we want to shift our focus to more task-specific analyses, as in previous analyses we found that the underlying network topology highly correlates to important performance metrics [4], [23]. Now that we investigated more general graph metrics and gained an overview of existing methods and their (dis)advantages, we may evaluate the similarity of communication network-centric performance indicators and investigate the desired variance for a specific use case.

Acknowledgment

This work has been funded by the German Federal Ministry of Education and Research (BMBF) as part of the project WINTERMUTE (16KIS1129).

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] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "SNDlib 1.0—survivable network design library," *Networks: An International Journal*, 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 *2020 International Conference on Information Networking (ICOIN)*. IEEE, 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 *2022 IEEE/IFIP Network Operations and Management Symposium (NOMS)*. IEEE, 2022.

[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 University 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," *Journal of 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 Transactions on Networking*, vol. 29, no. 6, pp. 2779–2792, 2021.

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

[11] 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.

[12] https://github.com/lsinfo3/cnsm2022-wan-topology-synthesis, [Online; accessed 30-August-2022].

[13] M. E. Newman and J. Park, "Why social networks are different from other types of networks," *Physical review E*, vol. 68, no. 3, p. 036122, 2003.

[14] J. M. Hernández and P. Van Mieghem, "Classification of graph metrics," *Delft University of Technology: Mekelweg, The Netherlands*, pp. 1–20, 2011.

[15] 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.

[16] F. Faez, Y. Ommi, M. S. Baghshah, and H. R. Rabiee, "Deep graph generators: A survey," *IEEE Access*, vol. 9, pp. 106 675–106 702, 2021.

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

[18] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, W. Li, X. Xie, and M. Guo, "Learning graph representation with generative adversarial nets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 8, pp. 3090–3103, 2019.

[19] 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 *Proceedings of the Web Conference 2021*, 2021, pp. 2104–2116.

[20] X. Guo, L. Wu, and L. Zhao, "Deep graph translation," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[21] S. Tavakoli, A. Hajibagheri, and G. Sukthankar, "Learning social graph topologies using generative adversarial neural networks," in *International Conference on Social Computing, Behavioral-Cultural Modeling & Prediction*, 2017.

[22] W. Liu, P.-Y. Chen, F. Yu, T. Suzumura, and G. Hu, "Learning graph topological features via GAN," *IEEE Access*, vol. 7, pp. 21 834–21 843, 2019.

[23] N. Gray, K. Dietz, and T. Hossfeld, "Simulative evaluation of KPIs in SDN for topology classification and performance prediction models," in *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 2020, pp. 1–9.

[24] https://github.com/lsinfo3/OpenFlowOMNeTSuite/tree/master/openflow/networks, [Online; accessed 02-May-2022].

[25] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[26] V. Zadorozhnyi and E. Yudin, "Growing network: Nonlinear extension of the barabasi-albert model," in *International Conference on Information Technologies and Mathematical Modelling*. Springer, 2014, pp. 432–439.

[27] 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 *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 388–396.

[28] https://github.com/tensorflow/docs/blob/master/site/en/tutorials/generative/dcgan.ipynb, [Online; accessed 17-June-2022].

[29] A. Grigorjew, M. Seufert, N. Wehner, J. Hofmann, and T. Hoßfeld, "ML-assisted latency assignments in time-sensitive networking," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2021, pp. 116–124.

[30] F. Loh, N. Mehling, S. Geissler, and T. Hoßfeld, "Graph-based gateway placement for better performance in LoRaWAN deployments," in *2022 20th Mediterranean Communication and Computer Networking Conference (MedComNet)*, 2022.

[31] M. J. Alenazi and E. K. Cetinkaya, "Resilient placement of SDN controllers exploiting disjoint paths," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, p. e3725, 2020.

[32] 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 *2017 29th International Teletraffic Congress (ITC 29)*, vol. 1. IEEE, 2017, pp. 152–160.

[33] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[34] 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 *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2018, pp. 1184–1188.

[35] M. Seufert, S. Lange, and T. Hoßfeld, "More than topology: Joint topology and attribute sampling and generation of social network graphs," *Computer Communications*, vol. 73, pp. 176–187, 2016.