IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 3, 2019, accepted February 5, 2019, date of publication February 11, 2019, date of current version March 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2898693

Learning Graph Topological Features via GAN

WEIYI LIU[1,2], PIN-YU CHEN[2], FUCAI YU[1], TOYOTARO SUZUMURA[2], AND GUANGMIN HU[1]
[1]School of Communication and Information Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[2]Big Data Analytic Group, IBM Watson Research Center, Yorktown Heights, NY 10598, USA

Corresponding author: Fucai Yu (fcyu@uestc.edu.cn)

This work was supported by the National Natural Science Foundation of China, under Grant 61571094.

ABSTRACT Inspired by the generation power of generative adversarial networks (GANs) in image domains, we introduce a novel hierarchical architecture for learning characteristic topological features from a single arbitrary input graph via GANs. The hierarchical architecture consisting of multiple GANs preserves both local and global topological features and automatically partitions the input graph into representative ''stages'' for feature learning. The stages facilitate reconstruction and can be used as indicators of the importance of the associated topological structures. The experiments show that our method produces subgraphs retaining a wide range of topological features, even in early reconstruction stages (unlike a single GAN, which cannot easily identify such features, let alone reconstruct the original graph). This paper is the firstline research on combining the use of GANs and graph topological analysis.

INDEX TERMS Generative adversarial nets, graph analysis, graph generation.

I. INTRODUCTION

Graphs have great versatility, able to represent complex systems with diverse relationships between objects and data. With the rise of social networking, and the importance of relational properties to the ''big data'' phenomenon, it has become increasingly important to develop ways to automatically identify key structures present in graph data. Identification of such structures is crucial in understanding how a social network forms, or in making predictions about future network behavior. To this end, a large number of graph analysis methods have been proposed to analyze the topology of the target network at the node [1], community [2], [3], and global levels [4], and perform certain data analysis and machine learning tasks.

Unfortunately, each level of analysis is greatly influenced by the underlying network topology, and so far no algorithm can be effectively and automatically adapted to arbitrary and complex network structures. For example, modularity-based community detection [5] works well for networks with separate clusters, whereas edge-based methods [6] prevail in dense networks. Similarly, when performing graph sampling, Random Walk (RW) is suitable for sampling paths [7], whereas Forrest Fire (FF) is useful for sampling clusters [8]. When it comes to graph generation, Watts-Strogatz (WS) graph models [9] can generate graphs with small world features, whereas Barabasi-Albert (BA) graph models [10] can simulate super hubs and regular nodes according to the scale-free features of the network.

However, real-world networks typically have multiple and potentially complex topological features, which may be beyond the expressive power of a graph model. Moreover, taking real-world networks into consideration also introduces another issue that traditional graph analysis methods have been struggling with: one may only have a single instance of a graph (e.g. the transaction graph for a particular bank), making it difficult to identify the key topological properties in the first place. In particular, many graph mining and analysis problems are often associated with both ''local topological features'' such as the presence of subgraph structures like triangles and ''global topological features'' such as degree distribution.

In this paper, we propose an unsupervised method, the Graph Topology Interpolater (GTI), to facilitate graph analysis with an aim of bypassing the aforementioned issues. GTI is a novel approach combining techniques from graph analysis and GAN based image processing techniques. In Figure 1, we demonstrate that naively feeding the full graph (here, a 20 node BA network [10]) into a standard GAN implementation (the DCGAN [11]) is unsuccessful; such a GAN structure is unable to learn to reproduce the original graph, and instead stuck in undesirable local minima.

Therefore, instead of directly and naively analyzing the entire topology of a graph as an image, GTI first divides the

The associate editor coordinating the review of this manuscript and approving it for publication was Zhong-Ke Gao.

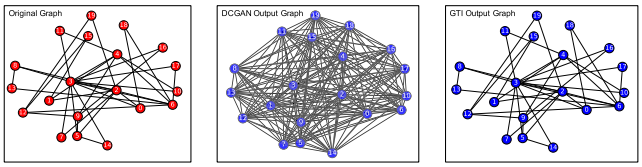


FIGURE 1. How GTI recovers the original graph while naive GAN methods fail.

graph into several hierarchical layers. A hierarchical view of a graph can split the graph by local and global topological features, giving a better understanding of the graph [12]. As different layers have different topological features, GTI uses separate GANs to learn each layer and the associated features. By leveraging GAN's renowned feature identification capability [13], [14] on each layer, GTI has the ability to automatically capture arbitrary topological features from a single input graph.

In addition to learning topological features from the input graph, the GTI method defines a reconstruction process for reproducing the original graph via a series of reconstruction stages (the number of which is automatically learned during training). As stages are ranked in the order of their contribution to the full topology of the original graph, early stages can be used as indicators of the most important topological features. This work is the firstline research in leveraging GAN to learn hierarchical topological features given a single input graph. We will demonstrate its ability to efficiently learn important topological features to retain critical structures and make comparisons to graph sampling methods.

Our proposed GTI method has three core contributions:

1) GTI is a model-agnostic graph topology analysis tool, which means it can analyze any arbitrary kinds of topology, rank the edge set, and outputs representative stages for this graph.
2) For each stage, it can preserve both local and global topological features of the input graph.
3) The input of GTI is only one single graph, which renders this method applicable to a wide range of applications with unique but rare graphs.

II. METHOD

In this section, we demonstrate the work flow of our Graph Topology Interpolater (GTI) (Figure 2), with a particular focus on the GAN, Sum-up and Stage Identification modules. At a high level, the GTI method takes an input graph, learns it's hierarchical layers, trains a separate GAN on each layer, and autonomously combines their output to reconstruct stages of the graph. Here we give a brief overview of each module.

Hierarchical Identification Module: This module detects the hierarchical structure of the original graph using the Louvain hierarchical community detection method [15]. Let L denote the number of layers. The average size of communities in each layer is used as a criterion for how many subgraphs a layer should pass to the next module.

Layer Partition Module: The main purpose of this module is to partition a given layer into M non-overlapping subgraphs, where M is the number of subgraphs. The reason why we do not use the learned communities from the Louvain method is that we cannot constrain the size of any community. We instead balance the communities into fixed size subgraphs using the METIS approach [16].

Layer GAN Module: Here we apply the GAN methodology to graph analysis. Rather than directly using one GAN to learn the whole graph, we use different GANs to learn features for each layer separately. If we use a single GAN to learn features for the whole graph, some topological features may be diluted or even ignored. See Section II-B for more details.

Layer Regeneration Module: Here, for a given layer, the corresponding GAN has learned all the properties of each subgraph, meaning we can use the generator in this GAN to regenerate the topology of the layer by generating M subgraphs consisting of k nodes. Note that this reconstruction only restores edges within each non-overlapping subgraph, and does not include edges between subgraphs.

All-layer Sum-up Module: By summing up the results from reconstructed layers, along with the edges that were not considered in the Regeneration Module, the purpose of this module is to output a weighted graph, where the "weight" represents the importance of an edge during the reconstructing process. Indeed, we rely upon these weights to identify the reconstruction stages for the original graph. For more details, see Section II-C.

Stage Identification Module: By analyzing the weighted adjacency matrix of the Sum-up Module, we can regenerate the graph by controlling different thresholds to generate an edge on a node-pair with respect to the value of the previously obtained weighted adjacency matrix. Here, we use the term "stage" to indicate a regenerate graph where the weights of edges are equal or larger than a given threshold. Note that these stages can be interpreted as steps for graph reconstruction, and can be automatically determined by the weights of edges. See Section II-D for details.

A. LAYER IDENTIFICATION AND LAYER PARTITION MODULE

In this module, we introduce a well-known hierarchical community detection method "Louvain" to identify the number of hierarchical layers in a graph [15]. By introducing a local optimization function "Incremental Modularity ΔQ," this approach calculates the ΔQ of a node and its neighbors, and selects one with the largest ΔQ. When $\Delta Q > 0$, the corresponding neighbor(s) are merged to the local community where the current node belongs to. The selection process will stop when local community in the graph is no longer changing. After that, Louvain method begins a new iteration by merging these local communities into a new layer in the hierarchy, and by re-using "finding maximum ΔQ" to detect new local communities in this layer. Consequently, this algorithm will automatically construct hierarchical layers for a graph.

As Louvain method helps to determine hierarchical structure and communities of the input graph in an unsupervised

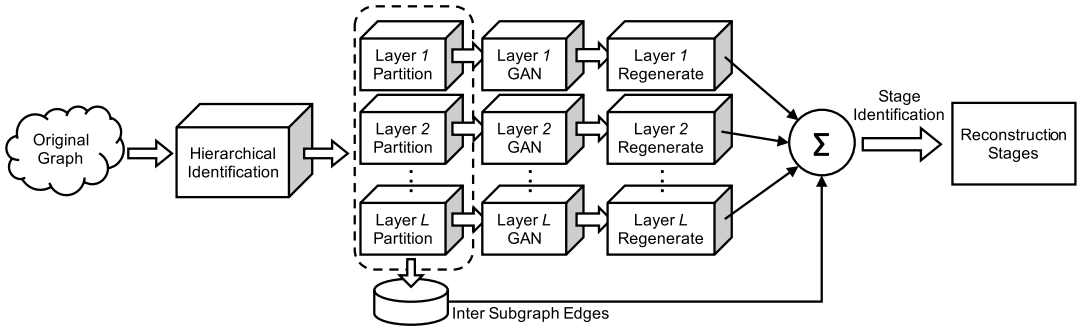


FIGURE 2. Work flow for graph topology interpolater (GTI).

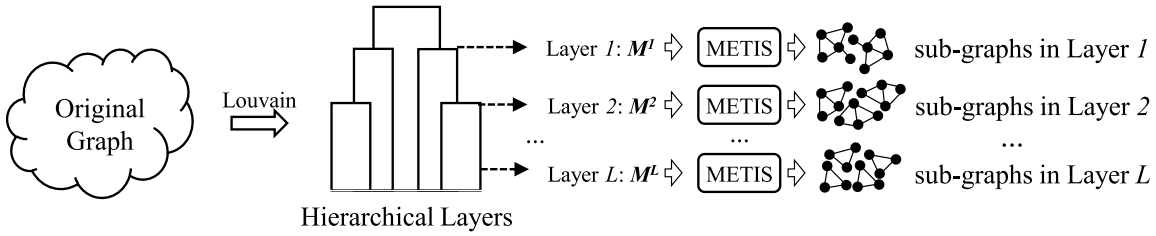


FIGURE 3. An example of Layer Identification and Layer Partition module.

manner, we use these two information – (a) number of hierarchies; (b) number of communities in the corresponding layer – as the prior information for guiding the algorithm to generate training samples for each layer. The reason to use the number of communities instead of communities themselves is because the input tensor for convolution neural network requires the same size of inputs. Hence, we need to unify the size of input samples for an generator to learn. In practice, this harsh requirement can be easily satisfied by adding isolated node(s) to the subgraph, or padding zeros to the adjacent matrix of the subgraph. We note that adding isolated nodes into a subgraph will definitely introduce extra noise. Nonetheless, we argue that ensuring a basically same subgraph size can maximally reduce the impact on the whole algorithm from the padding process. However, if we directly use the identified community as a subgraph, given the fact that one cannot ensure Louvain method will detect communities of the same size, one needs to add more isolated nodes to make same-sized communities and will inevitably incur unnecessary noise in the learning process. To tackle this challenge, we use METIS method to achieve this requirement. METIS method takes a graph as its input, and partitions the nodes in a balanced way so that each partition has the same number of nodes while minimizing the number of cut edges. Consequently, in the task of community detection with a known number of communities, METIS method finds same-sized communities with strong within-community connections.

The overall procedure on layer identification and layer partition module is summarized in Fig.3. First, by conducting Louvain method on the input graph, we can obtain the hierarchical structure. Here, each hierarchy stands for an layer,

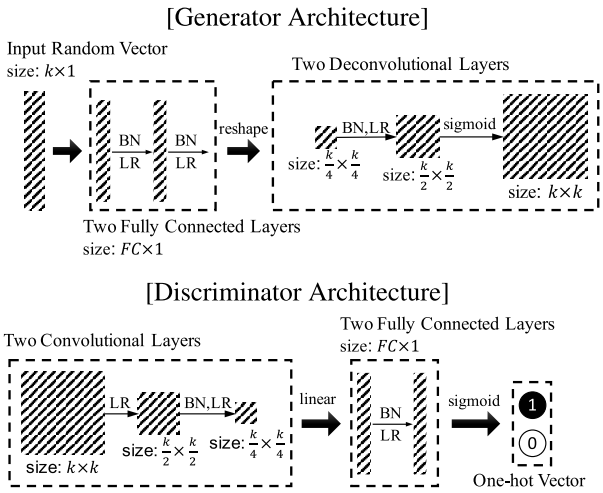


FIGURE 4. Generator and discriminator architecture.

and we can also obtain the number of communities in this layer (i.e. $M^1, M^2 \cdots M^l$ for layer 1, 2, and l, respectively). Then, for each layer, we use METIS method, along with the corresponding number of communities, to partition the original graph into same-sized communities (sub-graphs). We also note that higher layer in the hierarchy will have a larger community size.

B. LAYER GAN MODULE

Figure 4 shows the architectures for the generator and discriminator of the GAN. The generator is a deconvolutional neural network with the purpose of restoring a $k \times k$ adjacency matrix from the standard normal distribution.

The discriminator is instead a CNN whose purpose is to estimate if the input adjacency matrix is from a real dataset or from a generator. Here, *BN* is short for batch normalization which is used instead of max pooling because max pooling selects the maximum value in the feature map and ignores other values, whereas *BN* will synthesize all available information. *LR* stands for the leaky ReLU activation function ($LR = max(x, 0.2 \times x)$). Its value 0 carries a specific meaning for adjacency matrices (aka no edges). In addition, k represents the size of a subgraph, and *FC* is the length of a fully connected layer. We set the stride for the convolutional/deconvolutional layers to be 2. We adopt the same loss function and optimization strategy (1000 iterations of ADAM [17] with a learning rate of 0.0002), as used in DCGAN [11].

C. SUM-UP MODULE

In this module, we use a linear function (see Equation 1) to add the graphs from all layers together. re_G stands for the reconstructed adjacency matrix (with inputs from all layers), G'_i ($i \in L$) represents the reconstructed adjacency matrix for each layer (with G representing the full original graph with N nodes), E refers to all the inter-subgraph (community) edges identified by the Louvain method from each hierarchy, and b represents a bias. Note that while each layer of the reconstruction may lose certain edge information, summing up the hierarchical layers along with E will have the ability to reconstruct the entire graph:

$$re_G = \sum_{i=1}^{L} w_i G'_i + wE + b \tag{1}$$

To obtain the weight w for each layer and the bias b, we use Equation 2 as the loss function (where we add $\epsilon = 10^{-6}$ to avoid taking $log(0)$ or division by 0), use 500 iterations of SGD with learning rate 0.1 to minimize this loss function and find suitable parameters. We note that Equation 2 is similar to a *KL* divergence, though of course re_G and G are not probability distributions.

$$\text{Loss}\,(re_G, G) = \sum_{i \in 1 \cdots N^2} vec(G+\epsilon)_i \cdot log \frac{vec(G+\epsilon)_i}{vec(re_G+\epsilon)_i} \tag{2}$$

D. STAGE IDENTIFICATION

After the above calculations, we obtain the weighted adjacency matrix re_G, where weights on edges represent how each edge contributes to the entire topology. Clearly, different weights represent different degrees of contribution to the topology. Therefore, according to these weights, we can divide the network into several stages, with each stage representing a collection of edges greater than a certain weight. We introduce the concept of a ''cut-value'' to turn re_G into a binary adjacency matrix. We observe that many edges in re_G share the same weight, which implies these edges share the same importance. Furthermore, the number of unique weights can define different reconstruction stages, with the most important set of edges sharing the highest weight. Each stage will include edges with weights greater than or equal to the corresponding weight of that stage. Hence, we define an ordering of stages by decreasing weight, giving insight on how to reconstruct the original graph in terms of edge importance. We denote the ith largest unique weight-value as CV_i (for ''cut value'') and thereby denote the stages as in Equation 3 (an element-wise product), where $I[w \geq CV_i]$ is an indicator function for each weight being equal or larger than the CV_i:

$$re^i_G = re_G I[w \geq CV_i] \tag{3}$$

In Section IV, we use synthetic and real networks to show that each stage preserves identifiable topological features of the original graph during the graph reconstruction process. As each stage contains a subset of the original graphs edges, we can interpret each stage as a sub-sampling of the original graph. This allows us to compare with prominent graph sampling methodologies to emphasize our ability to retain important topological features.

By comparing GTI with some state-of-the-art graph sampling methodologies, we have found that the stages in GTI have similar performance for most metrics. Moreover, we note that since the number of stages in the reconstruction and the size of the graph in each stage are learned automatically, we do not have precise control over the size of the ''sampled'' graphs.

III. RELATED WORK

The development of deep learning and the growing maturity of graph topology analysis has led to more attention on the ability to use the former for the latter [18]. A number of supervised and semi-supervised learning method have been developed for graph analysis. A particular focus is on the use of CNNs [19]–[22]. These new methods have shown promising results on their respective tasks in comparison to traditional graph analysis methods (such as kernel-based methods, graph-based regularization techniques, etc). Kipf *et al.* has discussed the above methods in detail [22], [23], pointing out the strengths and drawbacks of various approaches. Recently, Sahar *et al.* [24] have proposed a naive method to generate the graph topology using GAN, by randomly perturbing the input graph 10,000 times, and feed these graphs into a DCGAN. It simply treats the adjacency matrix of a graph as an image, and uses random permutation to generate enough samples to train a DCGAN, without taking any topological information of the graph into consideration.

A key difference between GTI and other methods is that GTI is an unsupervised learning tool (facilitated by the use of GANs), that leverages the hierarchical structure of a graph. GTI can automatically capture both local and global topological features of a network. To the best of the authors' knowledge, this is the first unsupervised method in such manner.

Since GANs were first introduced [13] in 2014, its theory and application has expanded greatly. Many advances

TABLE 1. Size of original datasets, hyper-parameters for synthetic graphs, and corresponding reconstruction stages.

Graph	# Nodes	# Edges	# Stages	Retained edge percentage for ordered stages (%)
BA (m=2)	500	996	7	19.48, 26.31, 36.04, 39.36, 41.57, 57.43, 100
ER (p=0.2)	500	25103	4	4.32, 21.73, 94.91, 100
WS-1 (p=0.2, k=3)	500	500	7	11.20, 11.40, 16.00, 18.00, 54.60, 97.80, 100
WS-2 (p=0.2, k=6)	500	1500	8	15.60, 16.30, 21.04, 25.18, 38.62, 73.21, 98.13, 100
Kronecker	2178	25103	10	87.77, 88.65, 91.76, 91.89, 92.47, 93.32, 96.06, 97.05, 98.57, 100
Facebook	4039	88234	7	52.28, 83.33, 87.49, 91.41, 90.31, 91.95, 100
Wiki-Vote	7115	103689	4	58.31, 73.79, 85.60, 100
RoadNet	5371	7590	12	0.62, 3.87, 26.64, 27.98, 31.79, 32.42, 34.22, 34.65, 34.80, 64.06, 76.81, 100
P2P	3334	6627	7	49.04, 53.90, 70.32, 87.54, 88.40, 89.65, 100

in training methods [25]–[28] have been proposed in recent years, and this has facilitated their use in a number of applications. For example, GAN has been used for artwork synthesis [29], text classification [30], image-to-image translation [31], imitation of driver behavior [32], identification of cancers [33], etc. The GTI method expands the use of GANs into the graph topology analysis area.

IV. EVALUATION

All experiments in this paper were conducted locally on CPU using a Mac Book Pro with an Intel Core i7 2.5GHz processor and 16GB of 1600MHz RAM. Though this limits the size of our experiments in this preliminary work, the extensive GAN literature (see Section WorksIII) and the ability to parallelize GAN training based on hierarchical layers suggests that our method can be efficiently scaled to much larger systems. To study the benefit of the proposed GTI method, in this section we will also provide both qualitative and quantitative evaluations of the stages identified by GTI.

A. DATASETS

We use a combination of synthetic and real datasets. Through the use of synthetic datasets with particular topological properties, we are able to demonstrate the ability of retaining these easily identifiable properties across the reconstruction stages using GTI. In addition to validation on synthetic datasets with known topological properties, we also demonstrate our method on a number of real-world datasets with varying sizes.

We use the ER graph model [34], the BA graph model [10], the WS graph model [12] and the Kronecker graph model [35] to generate our synthetic graphs. Here, we have observed that for WS network, the hyper-parameter k will affect the clustering coefficient, as k controls the number of neighbors a node may connect in a WS network (WS-1). That is, if we set $k \leq 3$, the clustering coefficient for each node in WS network is 0. Hence, we add an extra WS network (WS-2) with $k = 6$ to avoid this case. For all synthetic networks, we use python package "*networkx-2.0*" to generate the corresponding graph.

For real datasets, we use data available from the Stanford Network Analysis Project (SNAP) [36]. In particular, we use the Facebook network, the wiki-Vote network, and the P2P-Gnutella network. The Facebook [37] dataset consists of "friends lists", collected from survey participants according to the connections between user-accounts on the online social network. It includes node features, circles, and ego networks; all of which has been anonymized by replacing the Facebook-internal ids. Wiki-vote [38] is a voting network (who votes for whom etc) that is used by Wikipedia to elect page administrators; P2P-Gnutella [39] is a peer-to-peer file-sharing network: Nodes represent hosts in the Gnutella network topology, with edges representing connections between the hosts. RoadNet [40] is the road network of Pennsylvania. Intersections and endpoints are represented by nodes, and the roads connecting them are edges. As this graph is of a size prohibitive to the computing resources used in work, we choose a connected component of appropriate size (note that the full network is not connected because nodes may be connected in the real-world by roads outside of Pennsylvania).

Detailed information for synthetic graphs as well as real-world datasets are outlined in Table 1.

B. LOCAL TOPOLOGICAL FEATURES

As discussed in Section II, GTI automatically ranks edge sets based on their contribution to the reconstruction of the original graph. Here we use two examples to demonstrate how this process retains important local topological structure during each reconstruction stage. By applying GTI to the BA network, the method learns that there are six stages for reconstruction of the original topology (with stages 1,3, and 5 shown in Figure 5). In our second example, we demonstrate GTI reconstruction for the real-world RoadNet dataset. Similarly, Figure 6 demonstrates the learned reconstruction stages 4, 8, and 11.

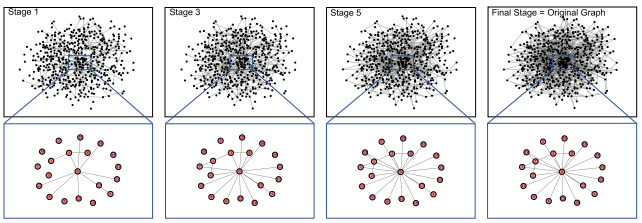


FIGURE 5. The topology of original graph and corresponding stages of BA network.

1) BA NETWORK STAGE ANALYSIS

We demonstrate the reconstruction process of a BA network in Figure 5, with the top row demonstrating the entire reconstruction process of the full network. We clearly observe that

each reconstructed network becomes denser and denser as additional stages are added. The bottom row of Figure 5 shows the subgraphs corresponding to nodes 0 to 19 at each reconstruction stage. We observe that these subgraphs retain the most important feature of the original subgraph (the star structures at node 0), even during the first reconstruction stage. In addition, we observe that the final stage exactly corresponds to the original stage. We again note that as illustrated in Figure 1, this reconstruction is extremely difficult when training a single GAN directly on the original graph.

2) ROAD NETWORK STAGES ANALYSIS

We observe in Table 1 that the retained edge percentages of the RoadNet reconstruction decrease more consistently with each stage than in the BA network. This is reasonable, because geographical distance constraints naturally result in fewer super hubs, with each node having less variability in its degree. In Figure 6, we observe the reconstruction of the full network, and the node 0 to node 19 subgraph of RoadNet. We observe in the bottom row of Figure 6 that the dominant cycle structure of the original node 0-19 subgraph clearly emerges.

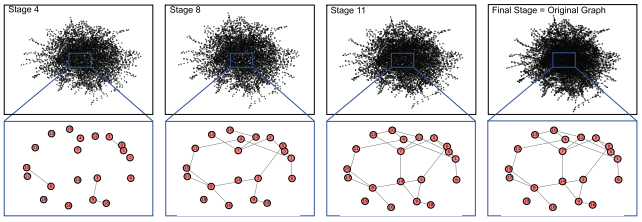


FIGURE 6. The topology of original graph and corresponding stages of road network.

We also observe an interesting property of the stages of the original graph in the top row of Figure 6. As SNAP does not provide the latitude and longitude of nodes, we cannot use physical locations. We instead calculate the modularity of each stage, where modularity represents how well connected the community is [41]. The tighter the community, the larger the modularity. We found that the modularity deceases from 0.98 to 0.92 approximately linearly. This suggests that the GTI stages first prioritize the clustering of nodes (through edge connections) over connections between clusters. This indicates that GTI views the dense connections between local neighborhoods as a particularly representative topological property of road networks.

C. GLOBAL TOPOLOGICAL FEATURES

In this section, we demonstrate the ability of GTI reconstruction stages to preserve global topological features, focusing on degree distribution and the distribution of cluster coefficients.

Figure 7 and Figure 8 respectively show the log-log degree distributions and log-log cluster coefficient distributions for each of the datasets given in Table 1. In Figure 7 (8), the horizontal axis in each degree distribution represents the ordered degrees (ordered cluster coefficient), with the vertical axis representing the density of each degree (the density of each cluster coefficient). For each sub-figure, we use a red line to demonstrate the (degree or cluster coefficient) distribution of the original graph, and use a set of colors (gradient from green to blue) to represent the corresponding distributions of the ordered stages.

We observe that with the exception of the ER network, the degree distributions and the cluster coefficient distributions of early stages are similar to the original graphs, and only become more accurate as we progress through the reconstruction stages. Although the degree distributions and cluster coefficient distributions for the early stages of the ER network reconstruction are shifted, we observe that GTI quickly learns the Poisson like shape in degree distribution, and also learns the ''peak-like'' shape in the cluster coefficient distribution. This is particularly noteworthy given that the ER model has no true underlying structure (as graphs are chosen uniformly at random). Finally, we note that the cluster coefficient for each node in WS-1 is zero, and we have observed that GTI learns this feature very quickly, even in the first few stages. However, as we cannot take the log of zero in the subplot, we plot the behavior of GTI on WS-2 instead in Figure 8. The results show that in both cases (zero or non-zero cluster coefficient), GTI is able to learn the corresponding characteristics.

In addition, we also use Frobenius norm [42] (see Equation 4) and average node-node similarity [43] (see Equation 5) to evaluate the similarity between generated stages and the original graph. Let $A = G - G'_{L-1}$, where G'_{L-1} is the adjacency matrix of the penultimate stage. The notation A^T denotes the matrix transpose of A, and $sim\left(n_i^{G'_{L-1}}, n_j^G\right)$ represents the node-node similarity with mismatch penalty [44] between node $i \in G'_{L-1}$ and node $j \in G$. $\left|N^G\right|$ is the total number of nodes in original graph. Here, F-norm calculates the distance between two adjacency matrices, and average node-node similarity indicates how G'_{L-1} resembles G.

$$F\_norm = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}^2} = \sqrt{trace(A^T A)} \tag{4}$$

$$sim(G'_{L-1}, G) = \frac{\sum_{i\in N}\sum_{j\in N} sim\left(n_i^{G'_{L-1}}, n_j^G\right)}{\left|N^G\right|} \tag{5}$$

For comparison, we use the same model parameters to generate 100 ensembles of BA, ER, Kronecker and WS networks. These newly generated networks serve as base models to help us evaluate the similarity between the penultimate stage and the original graph. Table 2 and 3 respectively display the F-norm and the average node-node similarity between the penultimate stage and original graph. Here, bold letters imply best values between the penultimate stage and the mean value of corresponding base models for each dataset. As for each base model, we have generated 100 samples. Hence, for

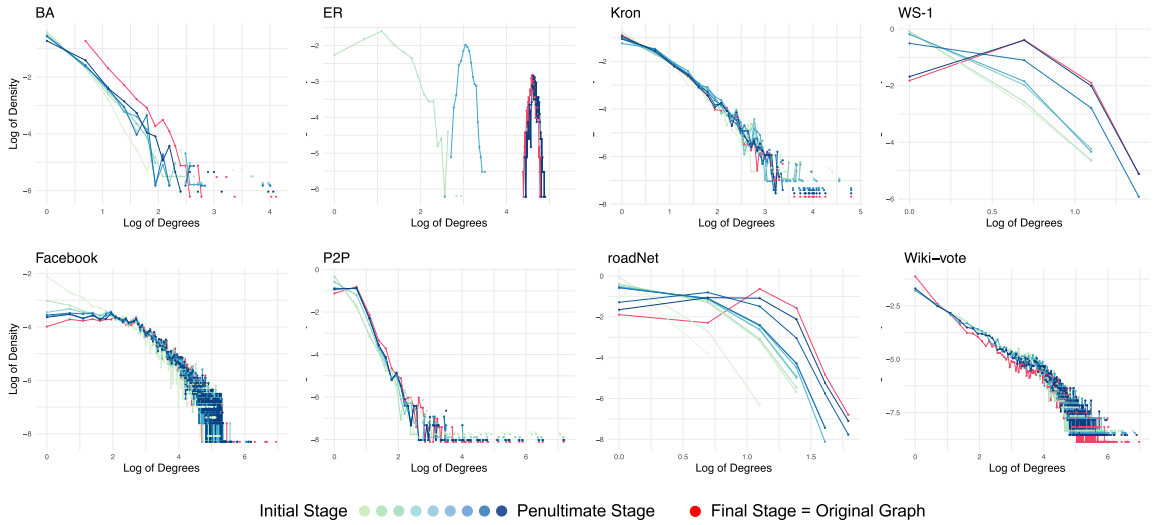


FIGURE 7. Degree distributions for 8 datasets.

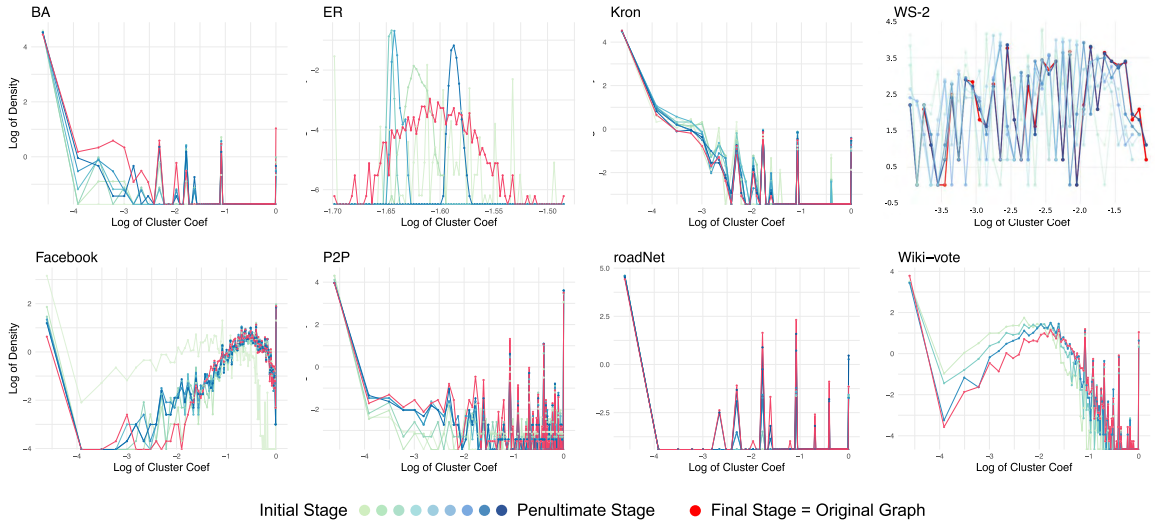


FIGURE 8. Cluster coefficient distributions for 8 datasets.

BaseModel and *Penultimate Stage* Column in Table 2 and 3, we add the standard error for each generated network.

- Table 2 shows that 3 of 4 penultimate stages (i.e. BA, Kronecker, WS) have smaller F-norm distance, which means GTI successfully maintains the topological information of the original graph.
- Table 3 shows that penultimate stage of BA has larger average node-node similarity with the original graph, and penultimate stages of Kronecker and WS and the corresponding base model have identical similarity to the original graph. These results give a solid evidence that GTI also has the ability to attain good node-level similarity to the original graph.
- The ER network is a totally random graph model and hence GTI performs slightly worse than base models (i.e., lack of important topological structure for GTI to learn).

D. COMPARISON WITH GRAPH SAMPLING

The graphs generated by GTI can be considered as samples of the original graph in the sense that they are representative subgraphs of a large input graph. We compare the performance of GTI with that of other widely used graph sampling algorithms (Random Walk, Forest Fire and Random Jump) with respect to the ability to retain topological structures [7]. We benchmark on the subgraph structures of the BA and Facebook datasets to compare the stage 1 of GTI against the graph sampling algorithms (designed to terminate with the same number of nodes as the GTI stage).

TABLE 2. F-norm distance numerical evaluation on penultimate stage and original graph.

Graph	Penultimate Stage	BaseModel	BaseModel_min	BaseModel_mean	BaseModel_max
BA	**53.0283**±0.0436	62.9031±0.0859	62.8172	62.9031	62.9762
ER	285.8566±0.6287	282.9252±0.5637	282.3615	**282.9252**	283.4431
WS-1	**44.4522**±0.0372	44.6094±0.0448	44.5646	44.6094	44.6542
WS-2	**44.3101**±0.0218	44.5103±0.0463	44.4640	44.5103	44.5566
kronecker	**124.9320**±0.2698	125.323±0.2757	125.1987	125.1987	125.5987

TABLE 3. Average node-node similarity numerical evaluation on penultimate stage and original graph.

Graph	Penultimate Stage	BaseModel	BaseModel_min	BaseModel_mean	BaseModel_max
BA	**99.9707%**±0.0047%	99.9640±0.0068%	99.9573%	99.9640%	99.9681%
ER	99.6372%±0.0196%	99.6590±0.0277%	99.6338%	**99.6590%**	99.6867%
WS-1	**99.9994%**±0.0001%	99.9994±0.0001%	99.9993%	**99.9994%**	99.9995%
WS-2	**99.9994%**±0.0001%	99.9994±0.0001%	99.9993%	**99.9994%**	99.9995%
kronecker	**99.2240%**±0.0593%	99.2241±0.0650%	99.1606%	**99.2240%**	99.2889%

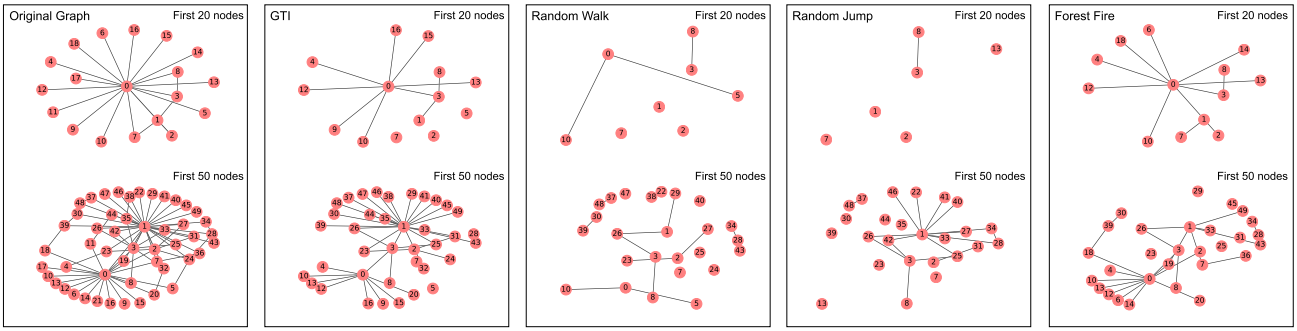


FIGURE 9. Comparison with graph sampling methods on the BA subgraphs.

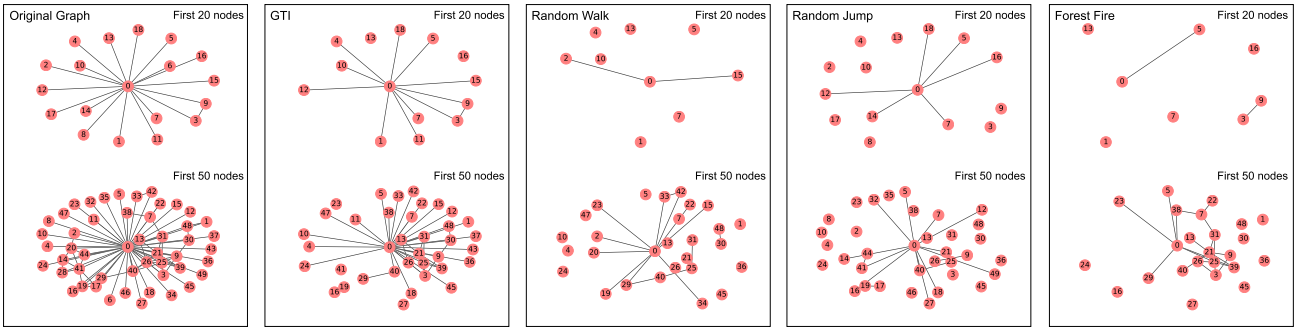


FIGURE 10. Comparison with graph sampling methods on the Facebook subgraphs.

To avoid messy graph plots, we show each of subgraphs from BA and Facebook networks (nodes 0-19 and nodes 0-49) to visually compare the ability of the first stage of GTI to retain topological features in comparison to the three graph sampling methods. In Figure 9, we observe that the stage 1 of GTI has retained a similar amount of structure in the 20 node BA subgraph as Forest Fire [8], while demonstrating considerably better retention than either Random Walk or Random Jump. However, for 50 node BA subgraph, only GTI has the ability to retain the two super hubs presenting the original graph. In Figure 10, we observe that GTI demonstrates vastly superior performance to the other methods on the Facebook dataset, which has a number of highly dense clusters with very sparse inter-cluster connections.

V. DISCUSSION AND FUTURE WORK

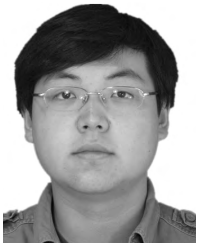
This paper aims to leverage the success of GANs in (unsupervised) image generation to tackle a fundamental challenge in graph topology analysis: a model-agnostic approach for learning graph topological features. By using a GAN for each hierarchical layer of the graph, our method allows us to effectively reconstruct input graph, preserving both local and global topological features. In addition,

our method is able to automatically learn the number of stages required to reconstruct stages and the graph itself non-parametrically. This is potentially advantageous in terms of understanding the distinct stages of graph reconstruction.

Our experimental results show promising results on the capability of GTI for learning distinct topological features from different graphs. To the best of our knowledge, there is not a single graph model that can capture these distinct topological features. A clear direction of future research is in extending the approach to allow the input graph to be directed and weighted, or with node attributions.

REFERENCES

[1] S. Muppidi and V. N. Koraganji, "Survey of contemporary ranking algorithms," *Int. J. Appl. Eng. Res.*, vol. 11, no. 1, pp. 322–325, 2016.
[2] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, nos. 3–5, pp. 75–174, 2010.
[3] V. Martínez, F. Berzal, and J.-C. Cubero, "A survey of link prediction in complex networks," *ACM Comput. Surv. (CSUR)*, vol. 49, no. 4, p. 69, 2016.
[4] Y. Wu, N. Cao, D. Archambault, Q. Shen, H. Qu, and W. Cui, "Evaluation of graph sampling: A visualization perspective," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 1, pp. 401–410, Jan. 2016.
[5] J. Xiang *et al.*, "Local modularity for community detection in complex networks," *Phys. A, Stat. Mech. Appl.*, vol. 443, pp. 451–459, Feb. 2016.
[6] A. Delis, A. Ntoulas, and P. Liakos, "Scalable link community detection: A local dispersion-aware approach," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2016, pp. 716–725.
[7] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2006, pp. 631–636.
[8] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: Densification laws, shrinking diameters and possible explanations," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2005, pp. 177–187.
[9] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.
[10] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.
[11] A. Radford, L. Metz, and S. Chintala. (2015). "Unsupervised representation learning with deep convolutional generative adversarial networks." [Online]. Available: https://arxiv.org/abs/1511.06434
[12] D. J. Watts, P. S. Dodds, and M. E. Newman, "Identity and search in social networks," *Science*, vol. 296, no. 5571, pp. 1302–1305, 2002.
[13] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. NIPS*, 2014, pp. 2672–2680.
[14] I. Goodfellow. (2016). "NIPS 2016 tutorial: Generative adversarial networks." [Online]. Available: https://arxiv.org/abs/1701.00160
[15] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Exp.*, vol. 2008, no. 10, 2008, Art. no. P10008.
[16] G. Karypis and V. Kumar, "Metis—Unstructured graph partitioning and sparse matrix ordering system, version 2.0," Dept. Comput. Sci., Univ. Minnesota, Minneapolis, MN, USA, Tech. Rep., 1995.
[17] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980
[18] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. (2015). "Gated graph sequence neural networks." [Online]. Available: https://arxiv.org/abs/1511.05493
[19] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Represent. (ICLR)*. Banff, AB, Canada: CBLS, Apr. 2014, pp. 3837–3845.
[20] M. Henaff, J. Bruna, and Y. LeCun. (2015). "Deep convolutional networks on graph-structured data." [Online]. Available: https://arxiv.org/abs/1506.05163
[21] D. K. Duvenaud *et al.*, "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2224–2232.
[22] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3837–3845.
[23] T. N. Kipf and M. Welling. (2016). "Semi-supervised classification with graph convolutional networks." [Online]. Available: https://arxiv.org/abs/1609.02907
[24] S. Tavakoli, A. Hajibagheri, and G. Sukthankar, "Learning social graph topologies using generative adversarial neural networks," Tech. Rep., 2017.
[25] E. L. Denton *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1486–1494.
[26] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Proc. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 2172–2180.
[27] J. Zhao, M. Mathieu, and Y. LeCun. (2016). "Energy-based generative adversarial network." [Online]. Available: https://arxiv.org/abs/1609.03126
[28] S. Nowozin, B. Cseke, and R. Tomioka, "f-GAN: Training generative neural samplers using variational divergence minimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 271–279.
[29] W. R. Tan, C. S. Chan, H. Aguirre, and K. Tanaka. (2017). "ArtGAN: Artwork synthesis with conditional categorical GANs." [Online]. Available: https://arxiv.org/abs/1702.03410
[30] T. Miyato, A. M. Dai, and I. Goodfellow. (2016). "Adversarial training methods for semi-supervised text classification." [Online]. Available: https://arxiv.org/abs/1605.07725
[31] Z. Yi, H. Zhang, P. Tan, and M. Gong. (2017). "DualGAN: Unsupervised dual learning for image-to-image translation." [Online]. Available: https://arxiv.org/abs/1704.02510
[32] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer. (2017). "Imitating driver behavior with generative adversarial networks." [Online]. Available: https://arxiv.org/abs/1701.06699
[33] S. Kohl *et al.* (2017). "Adversarial networks for the detection of aggressive prostate cancer." [Online]. Available: https://arxiv.org/abs/1702.08014
[34] Wikipedia. (2017) *Erdos/-Renyi/Model* [Online]. Available: https://en.wikipedia.org/wiki/ErdosRenyi_model
[35] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, "Kronecker graphs: An approach to modeling networks," *J. Mach. Learn. Res.*, vol. 11, pp. 985–1042, Feb. 2010.
[36] SNAP. (2017). *Stanford Network Analysis Project*. [Online]. Available: http://snap.stanford.edu/
[37] SNAP. (2012). *Social circles: Facebook*. [Online]. Available: http://snap.stanford.edu/data/egonets-Facebook.html
[38] SNAP. (2010). *Wikipedia Vote Network*. [Online]. Available: http://snap.stanford.edu/data/wiki-Vote.html
[39] SNAP. (2007). *Gnutella Peer-to-Peer Network*. Accessed: Aug. 4 2002. [Online]. Available: http://snap.stanford.edu/data/p2p-Gnutella04.html
[40] SNAP. (2009). *Pennsylvania Road Network*. [Online]. Available: http://snap.stanford.edu/data/roadNet-PA.html
[41] M. E. J. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci. United States Amer.*, vol. 103, no. 23, pp. 8577–8582, 2006.
[42] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, vol. 2. Philadelphia, PA, USA: SIAM, 2000.
[43] V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. Van Dooren, "A measure of similarity between graph vertices: Applications to synonym extraction and Web searching," *SIAM Rev.*, vol. 46, no. 4, pp. 647–666, 2004.
[44] M. Heymans and A. K. Singh, "Deriving phylogenetic trees from the similarity analysis of metabolic pathways," *Bioinformatics*, vol. 19, pp. i138–i146, Jul. 2003.



WEIYI LIU is currently pursuing the Ph.D. degree in communication and information systems with the University of Electronic Science and Technology of China (UESTC). His current interests include multilayer network embedding, community detection algorithms, and social network analysis. Currently, he is a recipient of the Chinese Government Scholarship, and has applied a two-year vising scholar with the IBM Watson Research Center, New York, NY, USA.

PIN-YU CHEN received the B.S. degree in electrical engineering and computer science (undergraduate honors program) from National Chiao Tung University, Taiwan, in 2009; the M.S. degree in communication engineering from National Taiwan University, Taiwan, in 2011; and the Ph.D. degree in electrical engineering and computer science, and the M.A. degree in statistics from the University of Michigan, Ann Arbor, MI, USA, in 2016. He is currently a Research Staff Member with the AI Foundations Learning Group, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. His recent research is on adversarial machine learning and robustness analysis of neural networks; moreover, his research interests include graph and network data analytics and their applications to data mining, machine learning, signal processing, and cyber security. He is a member of the Tau Beta Pi Honor Society and the Phi Kappa Phi Honor Society. He was a recipient of the Chia-Lun Lo Fellowship from the University of Michigan Ann Arbor. He received the NIPS 2017 Best Reviewer Award, and was also a recipient of the IEEE GLOBECOM 2010 GOLD Best Paper Award and several travel grants, including the IEEE ICASSP 2014 (NSF), the IEEE ICASSP 2015 (SPS), the IEEE Security and Privacy Symposium, the NSF Graph Signal Processing Workshop 2016, and the ACM KDD 2016.

FUCAI YU received the degree from the Lanzhou Railway College, in 1999, with a major in communication engineering, and the master's and Ph.D. degrees in computer communication and security from Chungnam University, South Korea, in 2006 and 2010, respectively. Since 2009, he has been with the School of Communication and Information Engineering, University of Electronic Science and Technology of China. In 2011, he was appointed as an Associate Professor by the University of Electronic Science and Technology. The main research directions include geographic location routing protocol, IP network traffic classification, backbone network anomaly event identification and analysis, and fault link diagnosis in sensor networks. At the current position, as the first author, he has published 6 SCI papers and 13 EI papers, such as the *Journal of Communications and Networks, IEICE Transactions on Communications*, IEEE Communications Letters, *IET Communications Magazine*, and IEEE ICC, Globecom, WCNC, and other well-known international conferences. The main courses are Exchange Principle and Communication Network Security.

TOYOTARO SUZUMURA is currently a Research Scientist, and also the Manager of the Graph Computing Department, IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY, USA. He has also served as a Visiting Full Professor with the Barcelona Supercomputing Center, Spain, and as a Visiting Researcher with The University of Tokyo, Japan. In 2017, he has served as the Program Committee Chair for the largest Big Data Conference and the IEEE BigData 2017. He and his team have kept winning the world championship in supercomputing competition called Graph500, for seven times, since 2014.

GUANGMIN HU received the degree from the Department of Computer Science and Technology, Nanjing University, in 1986, and the master's and Ph.D. degrees from the Chengdu University of Technology, in 1992 and 2000, respectively. He was a Postdoctoral Researcher with the University of Electronic Science and Technology, from 2000 to 2003, and also a visiting scholar with The Hong Kong Polytechnic University, from 2002 to 2003. He is mainly involved in the research of computer communication networks and signal and information processing. As the Principal Researcher or the Person in Charge of the research project, he has undertaken more than 50 research projects, including major projects of the Natural Science Foundation of China, the National Natural Science Foundation of China, the National 973 Program, and the National Science and Technology Research Projects and various horizontal topics. He has been selected for the New Century Excellent Talents Program of the Ministry of Education. He is the backup candidate for the academic and technical leaders in Sichuan Province.