Usage of a Graph Neural Network for Large-Scale Network Performance Evaluation

Cen Wang
Photonic Transport Network Lab.
KDDI Research, Inc.
Saitama, Japan
ce-wang@kddi-research.jp

Noboru Yoshikane
Photonic Transport Network Lab.
KDDI Research, Inc.
Saitama, Japan
yoshikane@kddi-research.jp

Takehiro Tsuritani
Future Network Infrastructure Division
KDDI Research, Inc.
Saitama, Japan
tsuri@kddi-research.jp

***Abstract*—To quickly and accurately perform network evaluation, we propose a graph convolutional network-based performance evaluation method for ultralarge-scale networks. The learning results show that our method outperforms the fully connected network and convolutional neural network in the prediction error of the end-to-end latency and network throughput. In addition, we show that our method is significantly less time-consuming than traditional methods.**

***Index Terms*—optical network performance evaluation, topology flexibility, graph convolutional network, large-scale network**

I. Introduction

Traffic diversity requires an adaptive network topology. The reason is that a good topology can reduce the number of hops on a traffic path, the probability of congestion and queueing delays. In large-scale networks, this effect is even more pronounced. To improve the network performance in support of diverse traffic patterns (i.e., spatial and temporal traffic features), active studies on enabling network topological flexibility have occurred in recent years. For example, modern datacenter networks (DCNs) usually connect tens of thousands of servers, although traditional hierarchical DCN structures with a fixed topology cannot effectively match dynamic east-west traffic. Zhang et al. [1] proposed a flattened small-world DCN structure using optical circuit switching (OCS) and optical packet switching (OPS). Xia et al. [2] proposed a reconfigurable structure for a Clos-based DCN using a special converter switch (CS). Another instance is the global low-earth-orbit satellite (LEO) network. Such a network has a lattice-shaped topology with an ultralarge network diameter, which considerably increases the network latency and decreases the network throughput. Our previous work [3] introduced a transparent forwarding module onto the LEO by using OCS. The modified structure can also achieve topological reconstruction by reconfiguring the optical switch matrix on the LEO. However, before building a mobile network and a content delivery network, one should apply user statistics (e.g., population distribution) and previous traffic statistics (e.g., statistical network assessment process [4]) to infer/predict an approximate future traffic pattern for network planning. In the aforementioned use cases, network performance evaluation is necessary to obtain the best topology; that is, we should verify whether the network with a dedicated topology works well for a specific traffic pattern in terms of, e.g., the end-to-end latency and network throughput.

This work was partially funded by the Ministry of Internal Affairs and Communications, Japan, grant number JP MI00316.

Traditional network performance evaluation is complex. As for the end-to-end latency, we commonly acquire a traffic arrival model, which is a stochastic process; and use queueing theory to deduce a general calculation formula. In addition, for the network throughput, the linear programming (LP) model is often used. The number of constraints is proportional to the square of the number of network nodes. For a network with hundreds of nodes, a few tools can solve such a large LP model for network throughput, and it takes hours or days. However, a network with topological flexibility expects to use online evaluation methods so that it can adapt to current traffic patterns in time. Additionally, a fast and effective evaluation method can improve the efficiency of network planning. The machine learning method shows a great parameterization ability for very complex tasks, which inspired us to use a related method to generate a neural model for high-speed evaluation. Wang et al. [5] proposed a network performance (i.e., latency) scoring model based on a fully connected (FC) neural network. However, the basic FC has to serialize a network topology and a traffic pattern that are graphs (i.e., non-Euclidean data structure) and cannot effectively capture the features of graphs; thus, its learning effect is limited. Besides, only one evaluation term is insufficient. A recent graph neural network (GNN) [6] was proposed by Kipf et al. to fulfill the tasks related to graph-like data. Network topologies and traffic patterns are natural graph data structures; thus, in this paper, we propose an efficient evaluation method for large-scale networks based on GNNs. More specifically, we select the use case of the LEO network with 288 nodes to test our GNN model. The learning results show that our GNN model outperforms the FC and a convolutional neural network (CNN) in terms of the prediction error. Moreover, we compare the time consumption of the training of the GNN and traditional LP models. The results show that using a GNN can significantly accelerate the evaluation time.

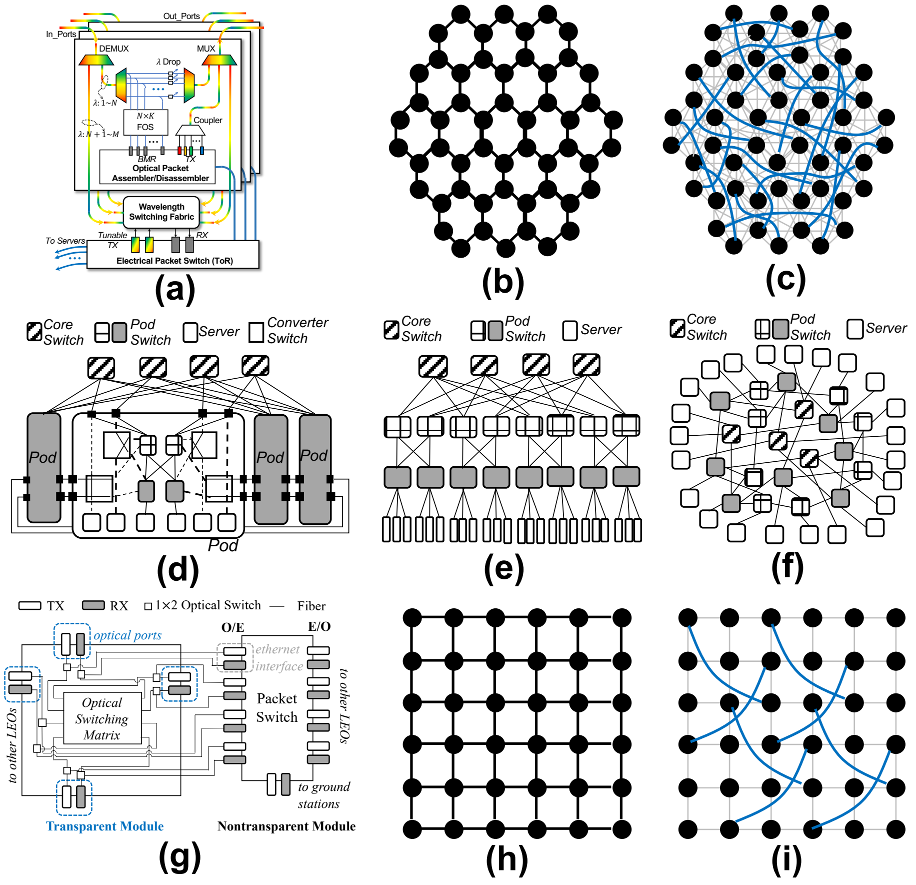

Fig. 1. The node/network architecture, basic/physical topology and reconfigurable logical topology of OpenScale ((a), (b) and (c)); Flat-tree ((d), (e) and (f)); and LEONet ((g), (h) and (i)).

II. Enabled Techniques

A. *Large-Scale Network with Topological Flexibility*

OpenScale [1]. The node structure is shown in Fig. 1 (a). Each node belongs to three adjacent OPS-based rings; thus, one node is implemented with three sets of input/output ports. For each input port, a DEMUX partitions wavelengths into two groups. One group connects the wavelength switching fabric to build the direct lightpaths, and the other enters the fast optical switch (FOS). The FOS selects the channels and forwards the drop traffic into the optical packet disassembler via burst mode receivers (BMRs). The additional traffic in each node can be switched either to the lightpaths (i.e., OCS) or to the optical packet assembler (i.e., OPS). The OpenScale network has multiple hexagonal rings composed of such OCS+OPS nodes, as shown in Fig. 1 (b). As shown by the blue paths in Fig. 1 (c), OpenScale uses OCS lightpaths to implement cross-ring connections without optical-electrical-optical (O-E-O) conversion, which is free of queueing delay and congestion. OPS, on the other hand, realizes intraring (i.e., time division multiplexing) meshed connections. As a result, the logical topology is similar to the Watts–Strogatz (WS) small world network [7].

Flat-tree [2]. As shown by the zoomed-in pod in Fig. 1 (d), in a pod of the flat-tree, an original pod-server link and an original pod-core link are disconnected, and the corresponding servers, pod switches, and core switches are connected to two 4-port CSs and two 6-port CSs. The 4-port CS has two alternative configurations. The "default" configuration maintains a basic Clos topology, and the "local" configuration enables the server to connect the core switch directly. The 6-port switch has two additional valid configurations: "side" and "cross". The "side" and "cross" configurations both relocate servers to core switches yet connect pod switches to their peers in different ways. To retain a Clos pod, the CSs and the additional wiring are packaged together. Flat-tree converts between multiple topologies with different CS configurations. Fig. 1 (e) shows the Clos network in which all CSs are with the "default" configuration. Fig. 1 (f) shows an approximate global random graph with the 4-port "local" and 6-port "side" configurations.

LEONet [3]. Each LEO is equipped with a switching system, and its principle is similar to that on the ground. One can forward Ethernet or IP packets using electrical packet switches (EPSs) according to the commercial standard digital video broadcasting (DVB) protocol. In Fig. 1 (g), we addi-

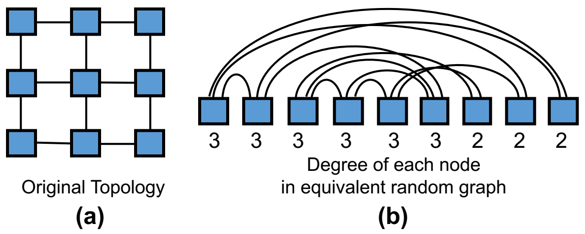

Fig. 2. How to transform (a) the original topology to (b) equivalent topology in order to calculate the relative network throughput.

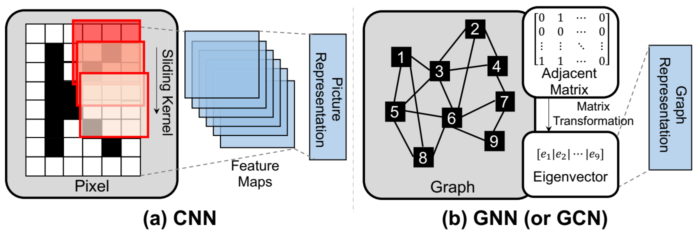

Fig. 3. The working principles of (a) a CNN and (b) a GNN.

tionally introduced an optical switching matrix on the node. The optical switching matrix has 4 optical interfaces (i.e., a pair of transmitters and receivers), and EPS has 8 optical interfaces and 1 RF interface (to the ground gate station). Four optical interfaces of the optical switching matrix and 4 optical interfaces of EPS are used to connect the adjacent LEOs via laser linking by spatial division multiplexing. Each port of the optical switching matrix connects to a 1×2 optical switch. By configuring the optical switch, the 4 optical interfaces can be redirected to the other 4 interfaces on the EPS so that the traffic can either transparently pass the node or go to the EPS domain for packet parsing. In this manner, over a basic LEO network topology with only one-hop intersatellite links (as shown in Fig. 1 (h)), optical lightpaths can be dynamically reconfigured between nonadjacent LEOs. Fig. 1 (i) shows a type of topology called "Knit" [3].

B. Large-Scale Network Performance Evaluation

End-to-end latency. For a network with a relatively large network diameter (the highest number of hops of all the paths), the end-to-end latency approximates the accumulation of the queueing latencies of all the nodes in a path [8].

We obtain a basic queueing delay $t_q^{(i)}$ of node i according to the queueing theory,

$$t_q^{(i)} = \frac{\rho_i}{1-\rho_i} \cdot \frac{1}{\mu_i} \tag{1}$$

where $\rho_i = \lambda_i/\mu_i$; and λ_i and μ_i are the traffic intensity (GB/s) and the node capacity (Gb/s), respectively. The node capacity is the number of ports $N_p^{(i)}$ times the port capacity C_p.

End-to-end latency is partially influenced by the routing policy. To obtain a general and valuable result, we set the routing policy to randomly select a path from all the shortest paths with equal probability. One can choose another preferred routing policy to evaluate. For a path p_j having n hops (i.e., the nodes), the end-to-end latency l_{e2e} is calculated by,

$$l_{e2e}(p_j) = l_p(p_j) + \sum_{i=1}^{n} l_q^{(i)} \tag{2}$$

where $p_j \in P$, and P is the set of all the paths of the network topology. $l_p(p_j)$ is the propagation delay of path p_j.

Network throughput. There are various definitions of the throughput. We adopt the definition of the network throughput as an evaluation term from Jyothi et al. [9]. If a network with topology A can accommodate k traffic matrix TM (the TM can be any), we call it the absolute network throughput. However, comparing the absolute network throughputs of different networks is unfair because a network with a larger node degree and a higher number of links certainly obtains higher throughputs. Thus, it is better to use a relative network throughput [9] (hereafter the network throughput) to avoid unfairness.

The relative network throughput uses the absolute network throughput of network A to divide the absolute network throughput of an equivalent network $E(A)$,

$$k_A/k_{E(A)} \tag{3}$$

where $E(A)$ is constructed according to A. The construction method (an example with a small number of nodes) is shown in Fig. 2:

- Add the degree of each node in the original network topology to obtain D_A.
- The equivalent network has the same number of nodes N as the original network. Divide D_A by N to obtain the quotient d_e and the remainder r_e.
- In the equivalent network, $N - r_e$ nodes are assigned d_e node degrees, and r_e nodes are assigned $d_e + 1$ node degrees.
- Randomly add edges between the nodes in the equivalent network until all the node degrees run out, and we obtain $E(A)$.

The LP model (a Java version of this linear program model can be found in [10]) that solves the network throughput takes the maximization of k as the objective, and the constraints include three parts: (1) the content (i.e., data size) in all links over the entire network that belongs to a traffic flow $TM(i,j)$ from source node i to target node j with data size $s_{TM(i,j)}$ does not exceed $k \times s_{TM(i,j)}$, (2) the sum of the data size from different traffic flows in a link does not cross the capacity of this link, and (3) the input and output traffic of a node should be equal.

C. Graph Neural Network

Neural networks learn the representations of input data to obtain target results. The basic CNN is shown in Fig. 3 (a), and it is suitable to process pixels. The CNN uses multiple channels of sliding kernels to perform 2D convolution and obtains

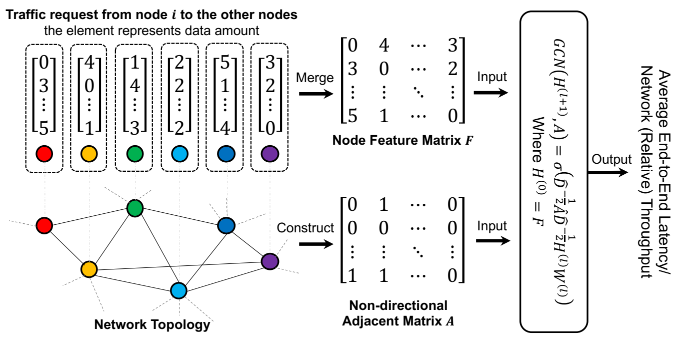
Traffic request from node i to the other nodes
the element represents data amount
Merge
Node Feature Matrix F
Input
Construct
Non-directional Adjacent Matrix A
Input
Network Topology
$GCN(H^{(l+1)}, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$
Where $H^{(0)} = F$
Output
Average End-to-End Latency/ Network (Relative) Throughput

Fig. 4. The usage of the GCN for network performance evaluation.

multiple feature maps. Then, pooling layers can transform the feature maps into the representations of the pictures.

In contrast, the input of the GNN is a graph, as shown in Fig. 3 (b). Although the popular GNN is called the graph convolutional network (GCN), no convolution operation is conducted. The internal processing of the GCN is a type of matrix transformation (spectral analysis). The GCN generates the eigenvectors of the graph and uses it to learn the graph representations.

The GCN can be described as (recursive form),

$$GCN(H^{(l+1)}, A) = \sigma(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}H^{(l)}W^{(l)}) \qquad (4)$$

where A is an adjacent matrix of a graph, and $H^{(l)}$ is the node (or vertex) feature matrix from the layer l. $\hat{A} = A + I$, where I is the identity matrix. $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$ and $W^{(l)}$ is a trainable weight matrix of the layer l. $\sigma(\cdot)$ denotes an activation function, such as $ReLU(\cdot) = max(0, \cdot)$.

III. Modeling for Network Performance Evaluation

The challenge of using a GCN is to map both the network topology and traffic pattern into the first layer of the GCN, that is, how to determine $H^{(0)}$ and A.

A. Modeling

We use a trick to regard the traffic matrix (in this matrix, an element at row i and column j denotes the amount of traffic data from i to j in a period of time) as the node feature matrix F, as shown in Fig. 4; that is, each column vector in F represents the received amount of data of each node (using a row vector is also acceptable). Then, we let $H^{(0)} = F$. The network topology is the matrix A, representing the node relationship in a graph. The GCN takes the traffic matrix and topology as inputs and transforms them into a predictive network performance term $\tilde{y}$ (i.e., $\tilde{k}$ and $\tilde{l}_{e2e}$) after an FC layer. By minimizing the mean square error (MSE) between y (i.e., k or l_{e2e}) and $\tilde{y}$, we can train a GCN to tune $W^{(l)}$ to achieve a lower MSE in the prediction of the average end-to-end latency and network throughput.

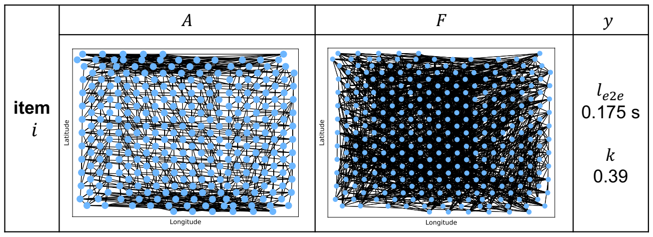

Fig. 5. An example item of the dataset.

TABLE I
PREDICTION OF MSE

Perc. of train data	25%	50%	75%	100%
Latency MSE	0.042	0.027	0.021	0.021
Throughput MSE	0.0144	8.22E-3	6.58E-3	6.49E-3

B. Dataset

We generate a dataset with 2000 data items (each item is one of the combinations of 50 topologies and 40 traffic patterns). A total of 1500 items were used for training, and the other 500 items were used for testing. Each item contains an adjacent matrix, a traffic matrix and a label (i.e., the latency or the throughput). When performing testing, the labels are removed. These labels were previously calculated by the aforementioned methods (Section II-B). An example item of the dataset is shown in Fig. 5, where the two matrices are visualized.

C. Learning Setup

Baselines. We take the basic three-layer FC and the basic three-layer CNN (the output layer is an FC) as the learning baselines. In the FC, we conjunct the adjacent matrix and feature matrix together and serialize the joint matrix (to a long vector) as the input. However, in the CNN, we regard the two matrices as two channels of input. The output of each baseline has 1 dimension (i.e., a value of latency or throughput).

The proposed GCN has 3 layers. In the input layer, the adjacent matrix and feature matrix are 288×288. The hidden size of the input layer and second layer is 16. The output layer is a fully connected network, and the output is also 1 dimension. We run the GCN for 500 epochs and retrieve prediction error every 20 epochs.

IV. Learning Result

Fig. 6 shows the comparisons of the testing MSE under the FC, CNN and GCN. The figure shows that for both the latency and throughput, the GCN converges well and achieves the lowest MSE. The CNN can converge but with some oscillation. The FC exhibits heavy oscillation and obtains unacceptably high errors. We also used different percentages of training data to test the model performance, as shown in Table I. The MSE is still acceptable when 50% of the training data are available. In addition, to show the prediction effects, we visualized the true value and the predicted values of the latency and throughput in Fig. 7 and Fig. 8, respectively. The figures show that the two network evaluation terms are sensitive to the

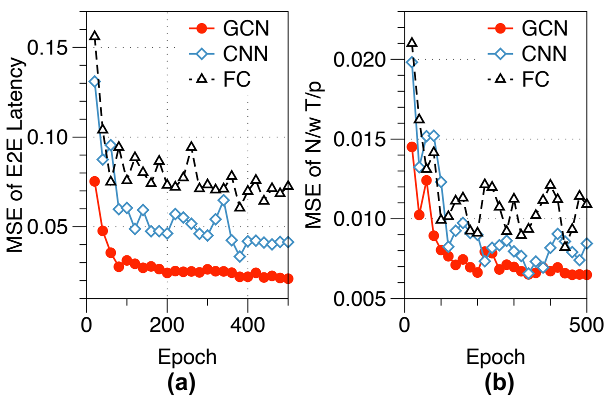

Fig. 6. MSE of (a) average end-to-end (E2E) latency and (b) (relative) network throughput in testing under a GCN, CNN and FC.

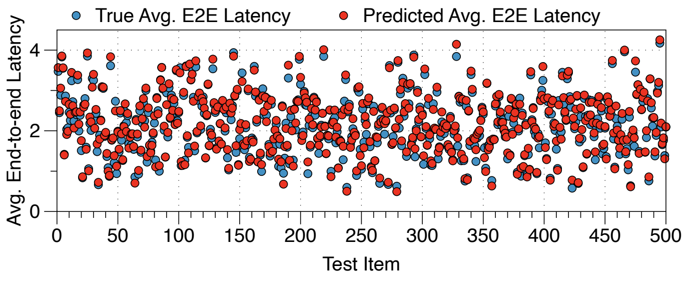

Fig. 7. Visualization of true and predicted average end-to-end (E2E) latency using the GCN.

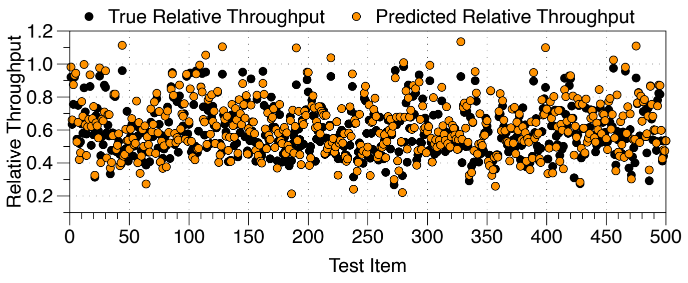

Fig. 8. Visualization of true and predicted (relative) network throughput using a GCN. (Note that the order of test items in this figure is different from that in Fig. 7 because we used "shuffle" when loading test data in two dependent learning processes.)

MSE. Even though the MSE of the throughput is lower than that of the latency, we can see that the fitting of the latency is better than that of the throughput.

The time consumption of the GCN (training) is 358.1 s on average. However, the time consumption of the method (i.e., the LP in Section II-B) to the calculate throughput ranges from 1672 s to 16982 s. This statistic suggests that our GCN is much faster than the traditional evaluation method.

Conclusion

We proposed a GCN-based, fast and effective network performance (average end-to-end latency and relative network throughput) evaluation model for a large-scale (optical) network with topological flexibility. The learning results verify that our GCN outperforms the basic CNN and FC in the learning MSE. Because the network performance terms are sensitive to the MSE, our GCN can achieve the best prediction effect. In addition, the time consumption statistic shows that our GCN can efficiently accelerate the evaluation compared to the LP-based throughput calculation. This work can benefit the topological reconfiguration algorithm and network planning.

References

[1] D. Zhang, H. Guo, T. Yang and J. Wu, J. "Optical switching based small-world data center network," in Computer Communications, 2017, vol. 103, pp. 153-164.

[2] Y. Xia, X. S. Sun, S. Dzinamarira, D. Wu, X. S. Huang and T. E. Ng, "A tale of two topologies: Exploring convertible data center network architectures with flat-tree," in Proceedings of SIGCOMM, August 2017, pp. 295-308.

[3] C. Wang, Y. Zhu, N. Yoshikane and T. Tsuritani, "Low earth orbit satellite network architecture with optical inter satellite links," in iPOP2020, Japan.

[4] M. Cantono and C. Vittorio, "Identifying and unlocking topological bottlenecks using SNAP and SDM solutions," in Proceedings of 2017 19th International Conference on Transparent Optical Networks (ICTON), IEEE, 2017, pp. 1-4.

[5] M. Wang, Y. Cui, S. Xiao, X. Wang, D. Yang, K. Chen and J. Zhu, "Neural network meets dcn: traffic-driven topology adaptation with deep learning," in Proceedings of the ACM on Measurement and Analysis of Computing Systems, vol. 2 no. 2, 2018, pp.1-25.

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

[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proceedings of International Conference on Learning Representations (ICLR), 2017.

[8] A. Erramilli, O. Narayan, and W. Willinger, "Experimental queueing analysis with long-range dependent packet traffic," in IEEE/ACM Transactions on Networking, vol. 4 no.2, 1996, pp. 209-223.

[9] S. A. Jyothi, A. Singla, P. B. Godfrey and A. Kolla, "Measuring and understanding throughput of network topologies," in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, November 2016, IEEE, pp. 761-772.

[10] https://github.com/ankitsingla/topobench/blob/master/README.