A Study on Inferring Communication Delays using Graph Convolutional Networks with Semi-Supervised Learning

Taisei Suzuki [1] Yuichi Yasuda [2] Ryo Nakamura [2] Hiroyuki Ohsaki [2]

Department of Informatics, School of Science and Technology, Kwansei Gakuin University, Japan [1]
Department of Informatics, Graduate School of Science and Technology, Kwansei Gakuin University, Japan [2]

1 Introduction

In large-scale communication networks consisting of many end hosts and routers, accurate acquisition, measurement, and estimation of communication delays between node pairs are essential for providing high-quality communication services.

In several QoS-related performance metrics — metrics for efficiency (throughput, communication delay/jitter, and response time), metrics for availability, metrics for reliability (loss rate and bit error rate) —, communication delay is one of the key metrics to realize several traffic control mechanisms.

Conventional instrumentation and measurement techniques are suitable when the size of the network to be measured is relatively small, or when the number of node pairs to be measured is relatively small. However, in evolving and complex networks, it is not trivial to acquire, measure, and estimate the communication quality at a huge number of routers and end hosts.

In this paper, as an initial step toward the realization of estimating communication quality (especially communication delays between node pairs) in a large-scale network, we investigate the potential of graph neural networks [1] with semi-supervised learning for estimating communication delays between node pairs. The major difference between our approach and that in [2] is in the type and the structure of the neural network. Namely, we use graph convolutional networks whereas authors of [2] use normal feed-forward neural networks.

2 Problem Formulation

The communication delay estimation problem studied in this paper is a problem of inferring communication delays between one reference node and all other nodes in the network only from measured communication delays at some nodes.

We assume that the topology can be represented by undirected graph $G = (V, E)$ and that the topology is known. However, it is assumed that information in the network other than the topological structure (e.g., propagation delay and bandwidth) are unknown.

We also assume that messages are routed in the network based on a specific routing protocol but neither the details of the routing protocol nor routing tables at routers are known.

The objective of this paper is to estimate communication delays $D_v(v \in V \setminus \{s\})$ between the reference node s in the network and node v.

Communication delays D_v are observable at specific nodes $V_O \subset V$ (measurement nodes). Communication delays D_v at all other nodes $V_U \equiv V \setminus V_O \cup \{s\}$ are unknown, which need to be estimated as accurately as possible.

The communication delay estimation problem is a minimization problem of errors in communication delays between the reference node and every unknown node. Thus, for instance, using the mean absolute error as a cost function, the communication delay estimation problem can be formulated as

$$\min_{\widetilde{D_v}, v \in V_U} E[|D_v - \widetilde{D_v}|], \tag{1}$$

where $\widetilde{D_v}$ is the estimated communication delay from the reference node to node v.

3 Communication Delay Estimation with Graph Convolutional Networks

Our neural network consists of three-layer GCNs coupled with a rectified linear unit (ReLU) and the classifier at the output layer with the logarithmic softmax function. The inputs to GCNs are node features (i.e., the reference node indicator and the degree of the node) and the output from the GCNs are communication delays from the reference node to all other nodes.

Communication delays are numeric, so it is possible to design a neural network for communication delay regression. However, in this paper, we classify communication delays into one of the multiple classes according to the communication delay so that GCNs are designed as a classifier of communication delays.

4 Experiment

In this section, we investigate how accurately communication delays between the reference node and all other nodes can be estimated using our GCN-based neural network.

4.1 Case without Queuing Delays

First, we intentionally use a rather simple experiment setup; graph G is an undirected and unweighted, and processing delays and queuing delay at nodes are all zero. Therefore, communication delays between the reference node and other nodes are equivalent to the numbers of hops between the reference node and other nodes.

For given network size N and density k (the average degree), undirected and unweighted graph G is randomly generated using ER (Erdõs-Rényi) model.

The single reference node is randomly chosen from all nodes in graph G.

Paths from the reference node to all other nodes in graph G is determined by the shortest-path routing. In our first experiment, the estimated communication delay corresponds to the number of hops, so the estimated communication delay $\widetilde{D_v}$ is equivalent to the shortest path length from the reference node to the node.

For given training set size ϕ, a fraction ϕ of nodes V_O are randomly chosen from all nodes excluding the reference node.

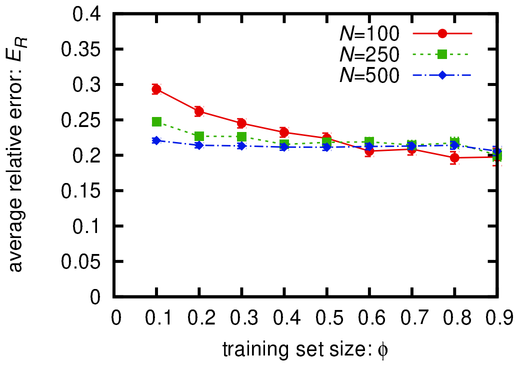

Figure. 1 Estimation accuracy of communication delays (ER model, $k = 3$)

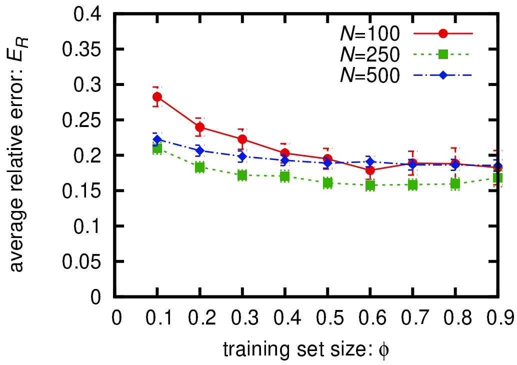

Figure. 2 Estimation accuracy of communication delays in the case of finite bandwidth (ER model, $k = 3$)

In all experiments, our GCNs were configured as follows: the number of features at intermediate GCN layers is equally set to 10, all weight parameters of GCNs are randomly initialized using the normal distribution with the mean of 0.2 and the standard deviation of 0.01. All bias parameters of GCNs are initialized to zero. We used the Adam (adaptive moment estimation) algorithm to update all weight and bias parameters of GCNs. The output from our GCN is a classifier with D_{max} classes where D_{max} is the maximum communication delay (e.g., the number of hops) in all observations (i.e., $D_{max} = \max_{v \in V \setminus \{s\}} D_v$). For given graph G and training set (i.e., a set of communication delays of measurement nodes), GCNs were trained for at most 5,000 epochs. We used the negative log likelihood loss as the loss function and the learning rate at each epoch of 0.001.

The estimation accuracy is measured by *the average relative error*. The relative error E_R is defined as follows.

$$E_R \equiv \frac{1}{|V_U|} \sum_{v \in V_U} \frac{|D_v - \widetilde{D_v}|}{D_v} \tag{2}$$

For given network size N, we randomly generated 500 network instances. With those 500 network instances, we measured the average relative error while changing a fraction ϕ of measurement nodes. We calculated the mean and the 95% confidence interval of the average relative error for training set size ϕ.

The average relative error for graphs generated with the ER model is shown in Fig. 1. From these results, it is found that the average relative error of estimated communication delays (i.e., the number of hops) is around 10–35% depending on the fraction of the number of measurement nodes. Also, it is found that communication delays for large-scale networks (e.g., $N = 500$) can be estimated with a high accuracy even if the fraction ϕ of the number of measurement nodes is not so large.

4.2 Case with Queuing Delays

Next, we use a more realistic scenario than the previous case; the network is composed of a large number of TCP end hosts, RED (Random Early Detection) routers, and the links with the finite bandwidth and the propagation delay, whose topology is given by a random network generated by the ER model.

Similarly to Section 4.1, for given network size N and the density k, the network topology composed of RED routers is randomly generated with the ER model.

The TCP sender is attached to the reference node, which is randomly chosen from all RED routers in the network. All other routers are attached with TCP receivers. The TCP sender continuously transmits data to all TCP receivers; i.e., the network accommodates $N-1$ persistent TCP flows from the reference node to all other nodes. For simulating background traffic, 40 persistent TCP flows are randomly placed in the network; i.e., 40 pairs of TCP sender and TCP receiver are randomly chosen from all RED routers in the network to accommodate persistent TCP flows.

For training and verification, all communication delays from the TCP sender to every TCP receiver are obtained using a fluid-based network simulator FSIM (Fluid-based SIMulator). The bandwidth and the propagation delay of all links are set to 50 [packet/ms] and 1 [ms], respectively. We used default values of FSIM for all other and TCP and RED control parameters.

As an additional input feature to GCNs, we used the number of hops from the TCP sender to every TCP receiver. Different from Section 4.1, for given network size N and density k, 200 network instances are randomly generated with the ER model.

Figure 2 shows the estimation accuracy of communication delays from TCP sender to TCP receivers. From these results, it is found that our GCN-based neural network can accurately estimate communication delays including queuing delays from TCP sender to TCP receivers under the realistic scenario. However, different from results in the case without queuing delays in Section 4.1, the estimation accuracy for a large-scale network (i.e., $N = 500$) does not improve even though the training set size increases.

Acknowledgments

This work was partly supported by JSPS KAKENHI Grant Number 19H04104 and 18J10278.

References

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*, pp. 1–14, Apr. 2017.

[2] F. S. Rizi, J. Schloetterer, and M. Granitzer, "Shortest path distance approximation using deep learning techniques," in *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2018)*, pp. 1007–1014, Aug. 2018.