Improved GNN Generalization to Larger 5G Networks By Fine-Tuning Predictions From Queueing Theory

Bruno Klaus de Aquino Afonso

PaRaNA - Pattern Recognition and Network Analysis group

Federal University of São Paulo (UNIFESP) - Brazil

1) Generalization on larger graphs

Possible approaches to predicting network delay

	Fast Enough?	Has top tier performance?	Generalizes to larger graphs?
Analytical	✓	✗	✓
Packet simulators	✗(prohibited)	✓	✓
RouteNet	✓	✓	✗
Proposed solution	✓	✓	✓

Table 1: Types of approaches

How can we generalize to larger graphs?

— We know that **the Analytic/Queueing Theory (QT) baseline is able to generalize well** to larger graphs

— We want features that are **invariant** w.r.t. to graph size. The baseline prediction is certainly that...

— Use Graph Neural Networks to **fine-tune** the baseline prediction

How can we generalize to larger graphs?

Raw path, link features:
Bad generalization

Raw features, divided by average number of packets generated (`p_AvgPktsLambda`):
Better generalization

Baseline features (path, link level prediction):
Best generalization

2) Model Architecture and Implementation

Implementation

- ➢ **Implemented from scratch in Pytorch + PyG**
    - Input had to be converted to **.pt** files

- ➢ **Two message-passing models:**
    - **Model 1:** Really big and takes longer to train
    - **Model 2:** Way smaller and achieves similar results
    - **Final submission:** Average of both!

Elements present in our model

- **Entities:**
  - Paths
  - Links
  - Nodes **(Model 1 only!)**
- **Types of Messages:**
  - Path-To-Link, Link-To-Path
  - Edge-To-Node, Node-To-Edge **(Model 1 only!)**
  - Path-To-Node, Node-To-Path **(Model 1 only!)**

Possible approaches to predicting network delay

Fixed Columns (contains initial features):

$$\mathbf{X}_P, \mathbf{X}_L, \mathbf{X}_N$$

Hidden state columns:

$$\mathbf{X}_{Ph}, \mathbf{X}_{Lh}, \mathbf{X}_{Nh}$$

Architectural differences v RouteNet

- Use **Graph Convolutional GRU** for Link-To-Path message passing
  - Links ordered according to path traversal
- The rest of the message passing uses **Graph Attention (GAT)** convolutions
- **Baseline features** are also kept unchanged during message passing rounds

Architectural differences

- **MLPs before and after** message passing
- Get **Link-level** predictions (avg. utilization), then:

$$\mathtt{delayLink}(i) = \mathtt{avg\_utilization}_i \times (\mathtt{queue\_size}_i / \mathtt{link\_capacity}_i)$$

$$\mathtt{pathDelay} \approx \sum_{i=0}^{\mathtt{n\_links}} \mathtt{delayLink}(i)$$

Algorithm 2 Model 2 (submitted on September 29th)

Require: $\boldsymbol{X}$ = `Concatenate([`$\boldsymbol{X}_P, \boldsymbol{X}_{Ph}, \boldsymbol{X}_L, \boldsymbol{X}_{Lh}, \boldsymbol{X}_N, \boldsymbol{X}_{Nh}$`],axis=1)`
Require: `baseline_path`, `baseline_link`: baseline predictions
Require: $\boldsymbol{E}$: network topology
Require: `NUM_iterations`: number of message-passing iterations
  `E_lp_list` ← `SeparateEdgeTimeSteps`($\boldsymbol{E}_{LP}$)
  $\boldsymbol{X}$ ← `MLP_1`($\boldsymbol{X}, \boldsymbol{E}_{LN}$)
  for $0 \leq i <$ `NUM_ITERATIONS` **do**
                                        ▷ Paths receive messages
    H ← `None`
    for $0 \leq k <$ `E_lp_list`.$length$ **do**
      H ← (`GConvGRU`$_{0,\texttt{link\_to\_path}}$($\boldsymbol{X}, H,$ `E_lp_list[k]`))
    end for
    $\boldsymbol{X}_{Ph}$ ← `LeakyRELU(H/(E_lp_list.length))`
    ($\boldsymbol{X}_{Ph}$)[:, 0:`baseline_path.shape[1]`] ← `baseline_path`
                                        ▷ Links receive messages
    $\boldsymbol{X}_{Lh}$ ← `LeakyRELU`(`Conv`$_{i,\texttt{path\_to\_link}}$($\boldsymbol{X}, \boldsymbol{E}_{PL}$))
    ($\boldsymbol{X}_{Lh}$)[:, 0:`baseline_link.shape[1]`] ← `baseline_link`
  end for
  $\boldsymbol{L}$ ← `Concatenate`(X_L, X_{Lh})
  $\boldsymbol{L}$ ← `Sigmoid`(`MLP_2`($\boldsymbol{L}$))   ▷ Predicts average queue utilization
  return `GetPathDelay`($\boldsymbol{L}, \boldsymbol{E}_{LP}$)   ▷ Obtains per-path-delay

Full report: https://github.com/brunoklaus/PARANA-GNNChallenge/blob/main/GNNET_2021_report.pdf

	# of hidden input columns	**MLP_1**	**MLP_2**
Model 1	$\boldsymbol{X}_{Ph}$:64 $\boldsymbol{X}_{Lh}$:64 $\boldsymbol{X}_{Nh}$:64	Seq(Linear(128), LeakyRELU(), Linear(inp_dim), LeakyRELU())	Seq(Linear(512), LeakyRELU(), Linear(512), LeakyRELU(), Linear(1))
Model 2	$\boldsymbol{X}_{Ph}$:8 $\boldsymbol{X}_{Lh}$:8	Seq(Linear(128), LeakyRELU(), Linear(inp_dim), LeakyRELU())	Seq(Linear(128), LeakyRELU(), Linear(32), LeakyRELU(), Linear(1))

	Baseline iterations	**Message passing iterations**
Model 1	5	3
Model 2	3	3

3) Results

Results

	Val. 1	Val. 2	Val. 3	Test
Model 1 (Sep 22nd)	2.71	1.33	1.65	1.45
Model 2 (Sep 29th)	3.61	1.17	1.55	1.45
(Model 1+ Model 2)/2	—	—	—	1.27
Baseline	12.10	9.18	9.51	?
Model 1 w/o baseline	—	—	—	22.58

-Also, using raw path/link metrics only and without division lead to >100% MAPE

-Using Baseline makes huge difference! (~5th to 1st place)

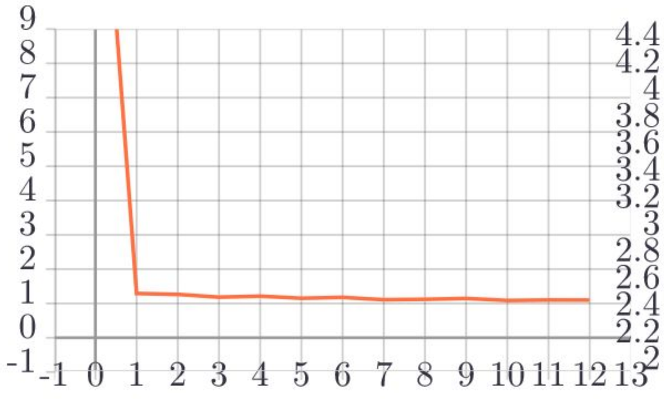

Figure 1: Training set

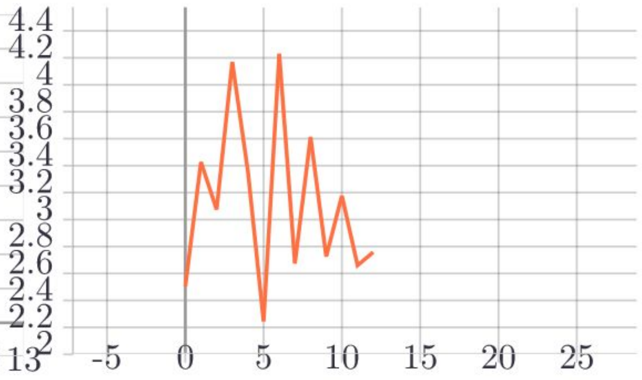

Figure 2: Validation set #1

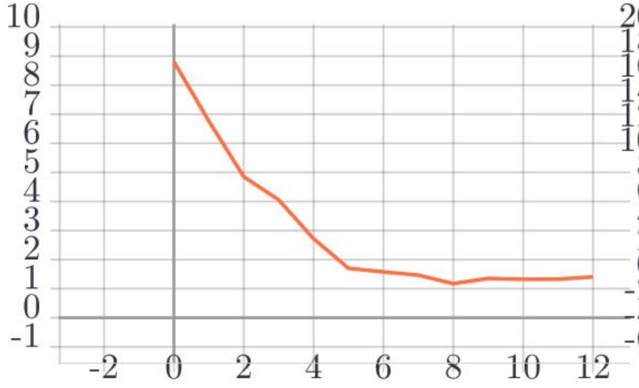

Figure 3: Validation set #2

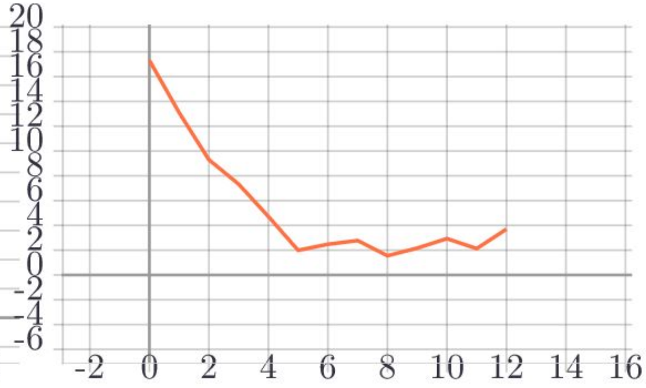

Figure 4: Validation set #3

References

[1] José Suárez-Varela et al. The graph neural networking challenge: a worldwide competition for education in ai/ml for networks. *ACM SIGCOMM Computer Communication Review*, 51(3):9–16, 2021.

[2] Krzysztof Rusek, José Suárez-Varela, Albert Mestres, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Unveiling the potential of graph neural networks for network modeling and optimization in sdn. In *Proceedings of the 2019 ACM Symposium on SDN Research*, pages 140–151, 2019.

[3] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[4] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[5] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*, pages 362–373. Springer, 2018.

[6] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, , Guzman Lopez, Nicolas Collignon, and Rik Sarkar. PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, 2021.

[7] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Thanks!

Any questions?

You can find me at:

- bruno.klaus@unifesp.br