ADDRESSING ROUTENET SCALABILITY THROUGH INPUT AND OUTPUT DESIGN

Junior Momo Ziazet[1], Charles Boudreau[1], Brigitte Jaumard[1,2], Huy Duong[2]
[1]Department of Computer Science and Software Engineering, Concordia University, Canada, [2]CRIM - Computer Research Institute of Montreal, Canada

NOTE: Corresponding author: Brigitte Jaumard, brigitte.jaumard@concordia.ca

Abstract – *With recent advances in the field of Machine Learning (ML), a multitude of problems related to communication systems and networks can be solved with data-driven solutions. Since data in these systems is mostly represented as graphs, Graph-based Neural Networks (GNNs) are a good candidate for solving such problems. These GNNs can be used as a computer network modeling technique to build models that accurately estimate the Key Performance Indicators (KPI) such as delay or jitter in real network scenarios in order to ensure their requirements in terms of service assurance. To build GNN solutions with higher accuracy, low computational resource requirements, and easy deployment of synthetic network training results into real-world networks, it is more than necessary to develop efficient and effective GNN models. This paper presents a GNN model capable of accurately estimating the average delay per flow in networks. By designing scale-independent features and using notions from queuing theory, the proposed model successfully generalizes to large size topologies, routing configurations, and traffic matrices not seen during the training phase.*

Keywords – 5G networks, graph neural network, KPI prediction, latency, network performance

1. INTRODUCTION

5G and B5G networks, which operate with Software Defined Networking (SDN) technology [1] have critical performance requirements from a network and user perspective. In particular, 5G/B5G networks offer service requests with very precise categorization, in order to allow network operators to design and operate dedicated slices for each service, while maintaining a certain level of both Quality of Service (QoS) and Quality of Experience (QoE) as perceived by the users [2]. These networks are expected to be very dynamic, and therefore require reliable methods to accurately predict expected KPI parameters to meet service assurance requirements [3]. To fulfill this objective, an important constraining factor to consider is that the proposed method must be scalable for larger topologies, which means that the solution must remain accurate when applied to much larger topologies than previously encountered by the solution. In the context of SDN, Rusek *et al.* [4] have shown that GNNs [5] are particularly promising for modeling computer networks. They presented a GNN-based model to predict key metrics to evaluate network performance. GNNs are designed to learn and model information structured as graphs and are able to capture complex interactions between network components and generalize to unseen network topologies. However, GNNs still have issues generalizing to networks whose scale is much larger than what the model has seen during training. In this paper, we aim to address this issue and we propose an efficient and scalable GNN model that can generalize well to unseen network topologies whose size greatly exceeds that of topologies that were encountered during the training process. We trained our model with samples simulated in topologies of 25 to 50 nodes, and we showed that the proposed model generalizes well with samples on larger network topologies with 50 to 300 nodes. We performed additional experiments to study the stability of our solution as well as its robustness.

This paper is structured as follows. In Section 2, we cover the context related to network performance estimation. In Section 3, we formally present the specific problem of estimating per-flow delay from given network statistics. In Section 4, we introduce graph neural networks and the RouteNet model on which our solution is based. In Section 5, we propose changes to the RouteNet model in order to address the scaling issues. We describe our experimental results in Section 6 and draw our conclusions in Section 7.

2. RELATED WORK

Traditionally, network calculus is a well-known theoretical framework for computer network analysis, including packet-based systems. However, network calculus requires the input of several bounds and distributions which are not realistic in practical systems, e.g., a lower bound on data forwarding or packet generation distribution. Besides, the most prohibitive aspect of network calculus is that it requires enormous computational times to proceed. To reduce computational times, researchers leverage deep learn-

ing and machine learning to accelerate existing network calculus techniques. Recent advancements in machine learning help network calculus reduce computational times significantly while remaining highly accurate. In [6], given a data flow, the authors proposed the DeepTMA model to predict delay. The DeepTMA model is an improvement of Tandem Matching Analysis (TMA) using GNNs to reduce the number of analyzed tandem decompositions. DeepTMA also shows that its model trained on networks composed of 10 nodes is able to generalize to topologies never before encountered in testing with 1,000 nodes. The execution time of DeepTMA is less than one second for large networks. However, network calculus requires strong assumptions on traffic settings. Usually, these assumptions are not available or practical, so one can use simulation runs to estimate the characteristics of traffic flows. Unfortunately, simulation runs are not only resource-intensive but also instance-driven, i.e., one simulation result cannot give answers to different settings. Recent work uses machine learning to learn the behaviors of simulation runs. Then, they infer characteristics of different settings based on learned models. In [7, 4], the authors proposed RouteNet for this strategy. It is a novel network model using Graph Neural Networks (GNNs) that can understand the complex interactions between topology, routing, and input data in order to obtain accurate estimates of the packet delay distribution and loss for each source-to-destination flow. Moreover, RouteNet has shown its ability to precisely infer characteristics of input (e.g., topology) unseen in the training [8]. Especially, in [9], the authors have shown a novel integration of GNNs into a Deep Reinforcement Learning (DRL) framework to solve routing problems by which trained models are capable of inferring solutions for unseen testing topologies. Comprehensive machine learning approaches to graph combinatorial optimization problems, including GNN design for several algorithms are presented in [10].

3. PROBLEM DESCRIPTION

Consider a physical network represented by a directed graph $G = (V, L)$ where V is the set of nodes and L is the set of directed links. Let F be a set of streams/flows. We are also given a set of routing paths R, where each routing path is a sequence of links connecting a source to a destination. Each flow $f \in F$ is mainly characterized by:

- a source s_k to a destination d_k
- $r_k \in R$: sequence of links connecting the source to the destination (routing path)
- $AvgBw_k$: average bandwidth reserved for the flow
- $PktsGen_k$: packets generated per time unit (packets/time unit)
- a complete list of random distribution parameters used to generate traffic is presented in [11]

If these settings are put into a simulation (e.g., OMNet++ [12]), then it can accurately report metrics of connection quality (e.g., average delay per package). However, a simulation requires a long waiting time (order of hours). Thus, the problem is to estimate the characteristics of flows faster. Beyond that, the solution must also be able to "scale" and remain effective even when applied to topologies which may be much larger than anything encountered during the development or experimentation phase.

4. NETWORK MODELING WITH GNN

In this section, we briefly describe the background on GNNs and RouteNet which are key concepts to understand the contributions of this paper.

4.1 Background on GNN

A graph can be seen as a data structure consisting of two components: nodes, which represent a specific object or data point, and edges, which represent relations between two objects. A graph neural network [5] is a type of neural network that can be directly applied to graph-structured data. A GNN takes graph data as inputs and produces node-level embeddings that encode the surrounding graph context for the nodes, which can then be used to infer properties about the individual nodes or the entire graph. There are many different variations of GNNs, but at their core, most GNNs have these two basic operations: a message-passing scheme and an update function. The message-passing scheme determines how information about the nodes' state is communicated to its neighbors, and the update function determines how a node's state is updated using the messages aggregated from neighboring nodes.

4.2 Background on RouteNet

RouteNet [4] is a network model based on a Graph Neural Network (GNN) that is able to understand the relationship between topology, routing, and input traffic to produce estimates of some KPIs in communication networks (per-flow mean delay for our study). RouteNet uses the input data, i.e., network topologies, source-destination routing schemes and traffic, to build a new heterogeneous graph with two types of nodes underlining the circular dependencies between links and flows. First, one node of type "link" is created for every link in the initial network topology. Then, for every flow, a "flow" node is added. In this architecture, a "link" node is connected to all the flows that traverse it and a "flow" node is connected to all the links it crosses.
Based on message passing GNN, RouteNet exchanges infor-

mation encoded between "link" and "flow" nodes to generate "link" and "flow" states that are then used for the downstream task, i.e., performance metrics prediction. More precisely, link and flow features are fed to a neural network to generate the initial "link" and "flow" embeddings. Then, messages are iteratively exchanged to update these embeddings. First, the path embeddings are updated from the ordered embeddings of its links. This task is done by an RNN to take into account the order of the links followed by a Gated Recurrent Unit (GRU). Next, each link embedding is updated by aggregating the embeddings of all flows that use that link. To represent the increasing load on the links, this aggregation is done by a global sum function, followed by a Gated Recurrent Unit (GRU). These two updates are repeated T = 8 times for every sample, both during training and inference. After that, the final flow hidden states of the message passing iteration are then processed in another neural network to readout the expected performance metrics (i.e., average flow delay for our case).

As stated in the baseline paper [4], RouteNet supports an arbitrary number of nodes in a network and can generalize well to other topologies. However, it has some difficulty generalizing to larger graphs. For the evaluation dataset, the RouteNet authors reported that in its unmodified state, Routenet's delay estimation on topologies of 50 to 300 nodes, given topologies of 25 to 50 nodes as training, achieves a Mean Absolute Percentage Error (MAPE) score in the range of about 300%. Our goal was therefore to improve its ability to estimate delay on larger topologies than those seen in the training.

The changes we made to the original RouteNet implementation allowed us to achieve a MAPE of less than 1.5%.

5. METHODOLOGY

In this section, we detail the methodology used to design a GNN that can estimate the average delay of a network as a function of the network topology, the routing and the traffic matrix.

5.1 Overview of the proposed solution

Noting the limitations of RouteNet and based on network data analysis, we made two major updates to the original RouteNet model.

As a first modification, we proposed a new feature that we call 'link load' which indicates the percentage of use of a link at a given moment. It is computed as the ratio of the summation of the flows traversing the link by the link's transport capacity, or maximum allowable bandwidth. The goal here is to provide dynamic information that matches exactly what is happening in the network rather than just using a static, absolute value like the link transport capacity as it was done in the baseline RouteNet model.

This feature was later used to further improve our solution by designing two other derived features. The first derived feature is 'link load squared' (link load raised to the power two) and the second one is 'link load cubed' (link load raised to the power three). This was done in order to not only help GNNs to better extrapolate by manually adding non-linearities to our features [13], but also because, by doing so, less busy links would be more easily differentiated from busier ones.

The second change of significance we implemented pertains to the model's output. Originally, the RouteNet model attempts to directly predict the delay for the individual flows in the network. One issue with this approach is that the distribution of delay values in the training set differs significantly from the distribution found in the validation and test sets as shown in Fig. 1a.

In order to address this observation, we changed the predictive variable. Instead of predicting the delay, our model predicts the occupancy ratio of the outgoing queues, which have a range of possible values bound between 0 and 1 in both training and validation sets, and then uses these predicted values to estimate the delay through a post-processing step that relates the occupancy ratio to the delay. Fig. 1b presents the distribution of the queue occupancy in the training and validation sets. We can see that both sets have similar queue occupancy distribution. This second modification makes RouteNet much more robust to changes in input topology size since the occupancy ratio is topology size independent.

With the aforementioned changes, the proposed model is an ensemble model. It is based on the combination of the solutions of two models, both based on Routenet.

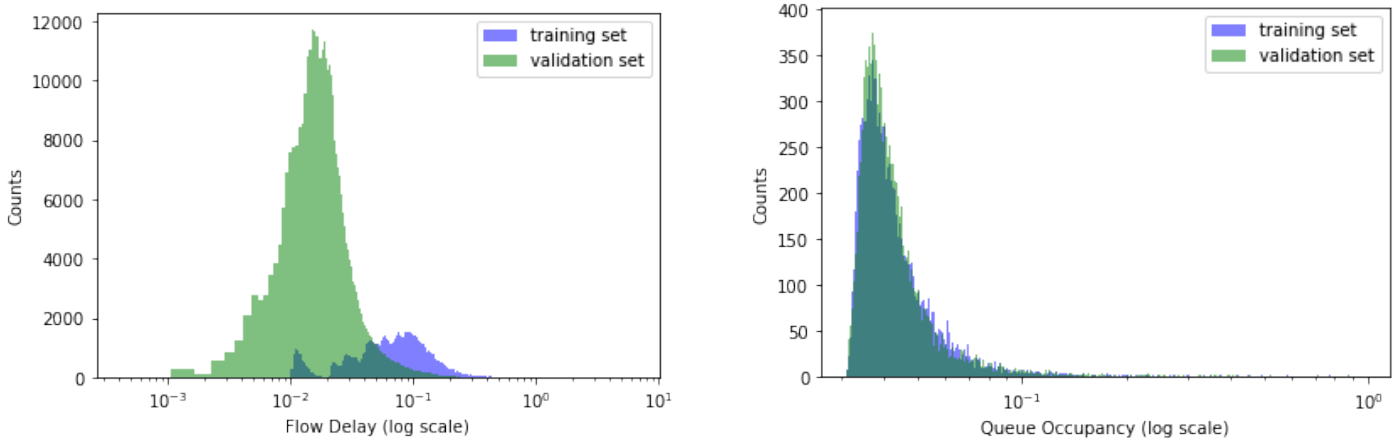
The following sections, 5.2, 5.3 and 5.4, describe in detail our contributions.

5.2 Proposed feature design

In order to provide GNN with a dynamic input to model the link feature, we designed a new link feature called "link load". This feature indicates in a relative way how busy a link is at a given time. The goal here was to provide the model with a characteristic that evolves consistently over time and remains in the same range ([0,1]) even if we change some network parameters such as link capacity. As presented in Eq. *(1)*, it is computed as the ratio of the summation of the flow traversing the link by the link's capacity, or maximum allowable bandwidth.

$$\text{Link\_Load} = \frac{\sum_{f \in N_\ell} t_f}{C_\ell}, \qquad (1)$$

where t_f is the value of the traffic feature of flow f, N_ℓ is the set of flows that traverse link ℓ, and C_ℓ indicates the transport capacity of the link (link bandwidth (bits / time unit)).



(a) Delay distribution on training and validation sets

(b) Queue occupancy on training and validation sets

Fig. 1 – Dataset distributions

According to Xu *et al.* [13], GNNs, as well as fully-connected neural networks using ReLU activation, can only fit linear functions outside of the training support. Encoding non-linearities either through the architecture of the model or, as in this case, through the features may result in the model only needing to fit a linear function and thus allowing it to generalize to unseen data more effectively. Based on that, we further improve our model by designing two more derived features by raising link load to the power 2 and 3 respectively called 'Link Load squared' and 'Link Load cubed'. This allowed us to add some non-linearities to the input that will help the GNN extrapolate. Futhermore, raising link load to the powers 2 and 3 would also provide features that show significant differentiation between links that are heavily used from those that are not.

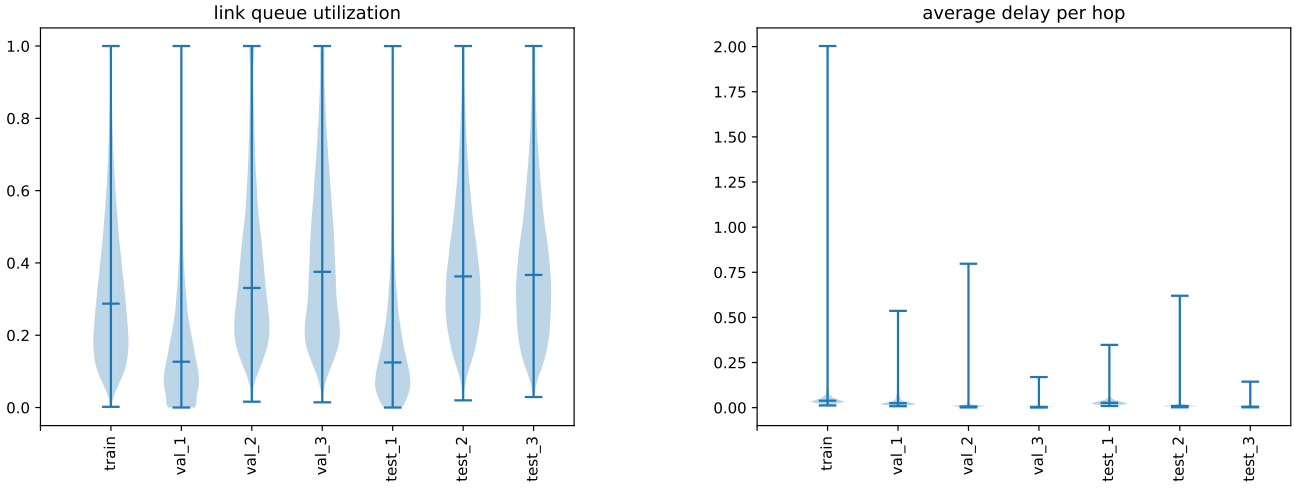
5.3 Proposed output transformation

The validation and test sets are both divided into three different settings, which are: setting-1 (longer paths than those found in the training set), setting-2 (larger link bandwidth capacity), and setting-3 (longer paths and larger link bandwidth capacity). In Fig. 2a and Fig. 2b, 'train', 'val_1', 'val_2', 'val_3', 'test_1', 'test_2' and 'test_3' represent the training, validation setting-1, validation setting-2, validation setting-3, testing setting-1, testing setting-2, and testing setting-3 sets respectively. In Fig. 2a, we report the queue utilization distributions for training, validation and testing data. It shows that the data contains a wide range of queue utilization rates, despite a low loss rate of the simulation setting (at most 3% loss rate). Thus, it implies that learning and prediction of queue utilization for a link of the data are not trivial. Fig. 2b recaps the average delay per hop of flows in the data. It shows that the ranges of average delay per hop of flows are narrow, especially in the test data. Thus, it suggests that predicting delay per hop is a promising direction for the data. Based on these insights from the data, we change the predictive variable of the model from delay to queue occupancy of the outgoing queues. This way, the scalability issue is indirectly partially solved since instead of a path-level prediction, where the path length can vary significantly when going to a small to a larger network topology, we predict a link-level measure that is not affected by the network size. By scalability issue, we mean that the accuracy of the path-level prediction gets worse when testing with topologies larger than those seen in the training data. Thus, the proposed solution predicts, for each link, the "queue occupancy", which is expressed as the average port occupancy (service and waiting queue) of the QoS queue (packets) divided by the queue size (packets). In other words, this is a normalization of the average port occupancy. We then compute the combined queue and propagation delay to obtain a delay value for a given link. In particular, path delays can be estimated as the linear combination of delays encountered in the queues of the outgoing links of the nodes along the path, considering the average Queue Occupancy (QO), the Queue Size (QS), the Packet Size (PS), and the Capacity (C) of the outgoing links of the queue. The link delay and flow delay expressions are represented below in Eq.*(2)* and Eq. *(3)* respectively

$$\text{Delay}_\ell = \text{QO}_\ell \times \text{QS}_\ell \times \text{PS}_\ell / C_\ell, \tag{2}$$

$$\text{Delay}_f = \sum_{k=1}^{N_f} \text{Delay}_\ell, \tag{3}$$

where N_f is the number of links that defines the flow path f, and C_ℓ the transport capacity of link ℓ.



(a) Queue utilization

(b) Average delay per hop

Fig. 2 – Dataset analysis per settings

5.4 Proposed GNN architecure

Since combining several models generally leads to a reduced bias and variance in the results, the solution we propose is an ensemble of two models that results in improving the score of each single model. Indeed, as shown in Fig. 3, our solution is an ensemble of two RouteNet based models (GNN_1 and GNN_2) where from the orignial RouteNet model, we modified the size of the link hidden state and path hidden state as well as the number of units of each DNN to adapt it to our designed features and output. While each of these models had a relative error higher than 1.4% on the validation set, averaging their output actually led to a lower error. Our best solution used the arithmetic mean of the output of the two models, leading to a MAPE of under 1.3%.

In our ensemble model, GNN_1 differs from GNN_2 solely through their respective inputs. While GNN_1 takes as input link features our designed link load and link load squared, GNN_2 has an additional feature which is link load cubed. With both models, after four rounds of message passing, the link hidden state will be fed to the readout function represented here with a neural network to predict the link queue occupancy. Then with a post-processing step represented in equations *(2)* and *(3)*, both models will output their predictions and the final prediction will be their average (arithmetic mean).

5.5 Features selection and preprocessing

From the dataset, we extracted the input features that are presented in Table 1 to be fed to each model in our solution. In Table 1, link_load, link_load_squared, link_load_cubed are our designed features described in Section 5.2. Furthermore, as defined in [11], "traffic" is the average bandwidth of a given traffic flow (bits/time unit), "packets" is the packets generated by a traffic flow per time unit (packets/time unit), "Eqlambda" is the time distribution feature (average bit rate per time unit (bits/time unit)). Flow input features show important differences in their range of values: for instance, traffic varies from 30.787 to 2048.23 while packets vary from 0.032895 to 2.03633 and Eqlambda from 40.0337 to 1999.52. Hence, we normalized them using Min-Max normalization.

6. EXPERIMENTS

We now present the experiments we conducted to evaluate the scalability capacity of our model. We explore different aspects, the benefit of our designed feature and output, the stability of our model as well as its robustness.

6.1 Dataset and settings

In this work, we used the dataset provided by "The Barcelona Neural Betworking Center" from the Universitat Politecnica de Catalunya. This dataset was created using OMNet++ [12], a packet-accurate network emulator. They comprise hundreds of route configurations and traffic matrices, as well as samples simulated in many topologies. The network performance metrics obtained by the simulator are labeled on each sample: per-source-destination performance measurements (mean per-packet delay, jitter, and loss), as well as port statistics (e.g., queue utilization, size). We had three separate datasets available to us: one for training, one for validation, and one for evaluation. The validation and test datasets contain samples of networks

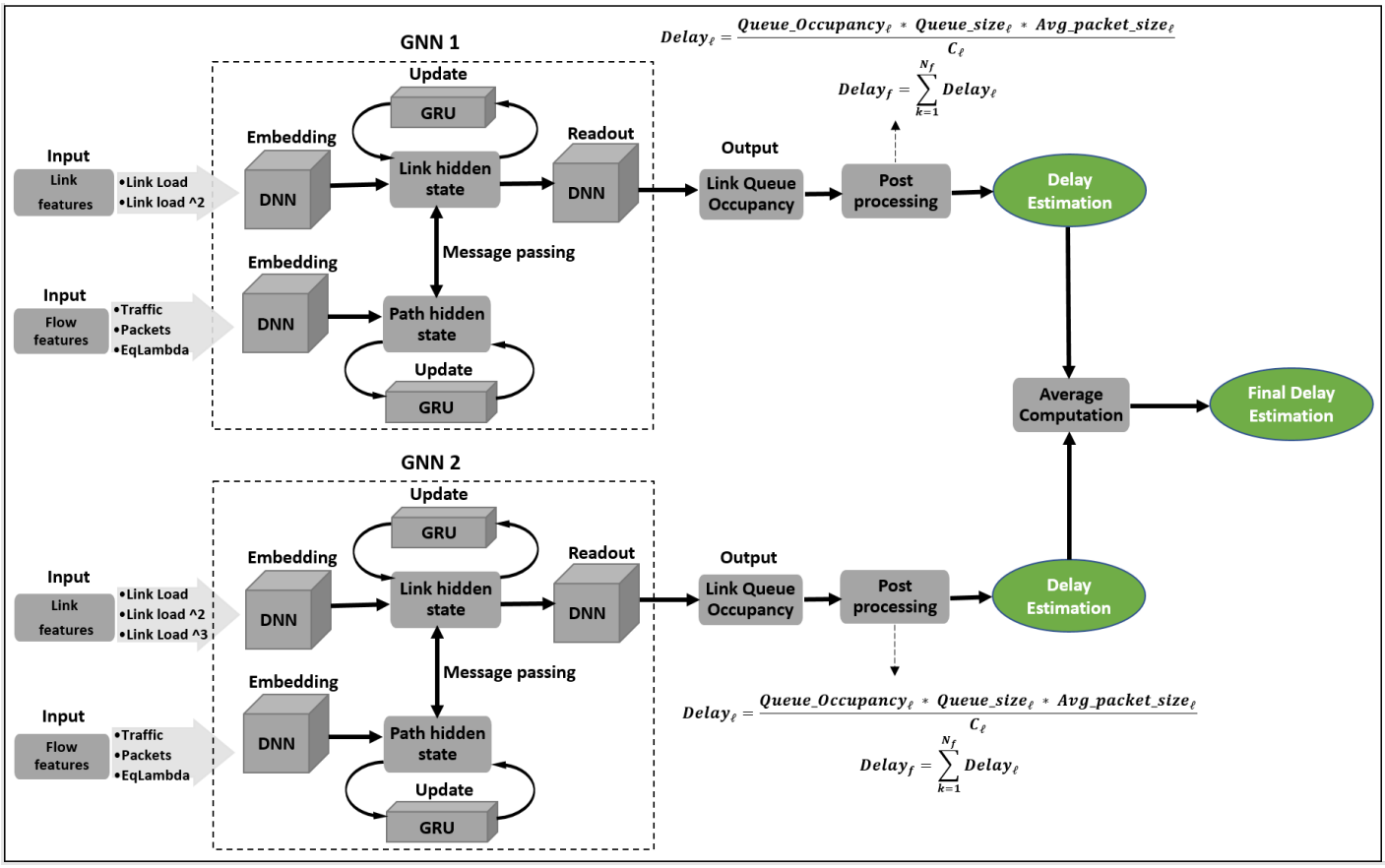


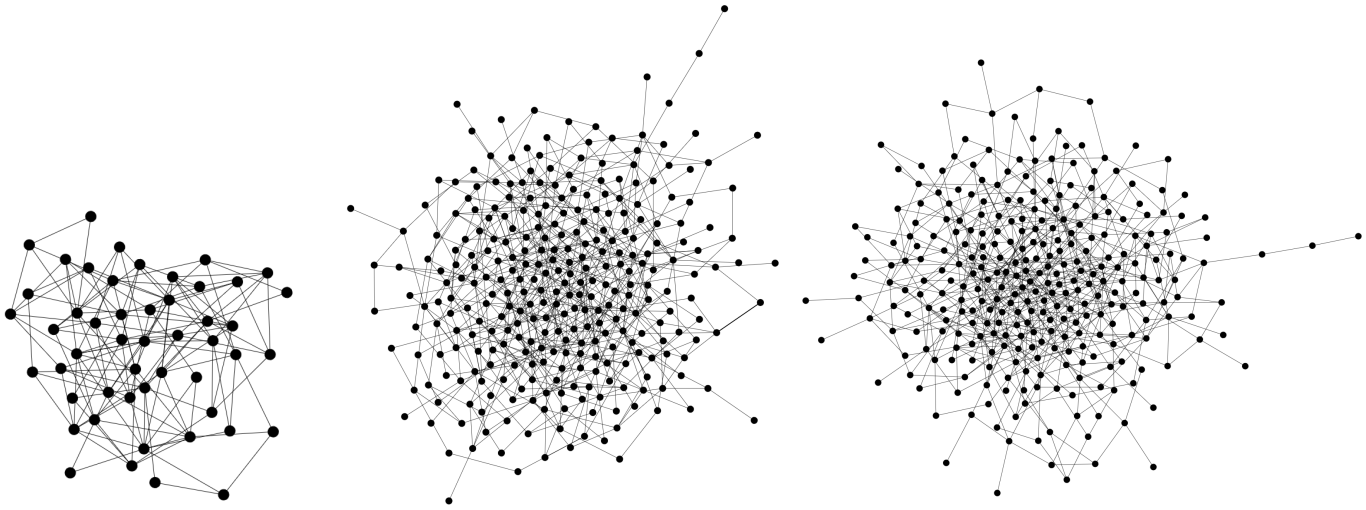
Fig. 3 – Architecture of our proposed solution

Table 1 – Model input features

	GNN 1	GNN 2
Flow features	Traffic	Traffic
	Packets	Packets
	EqLambda	EqLambda
Link features	Link Load	Link Load
	Link Load squared	Link Load squared
	-	Link Load cubed

that are significantly bigger (51-300 nodes) than those in the training dataset (25-50 nodes), as the work is focused on scalability. A detailed description of the dataset can be found on the GNN Challenge 2021 website[14]. In Fig. 4 and Fig. 5, we plot for a sample graph from the training, validation and test data set i) its topology shape, ii) number of flows per node pair, iii) total bandwidth requirement of the flows per node pair, and iv) total packet generation rate of the flows per node pair. First, we observe that, for these two samples, there is only one flow at each source-destination pair. However, because flows have different bandwidth and packet generation rates, we can consider the traffic flows heterogeneous. Besides, although the simulation is compatible with different 5G queue types (e.g., 'FIFO' (First-In, First-Out), Strict Priority, Weighted Fair Queuing, Deficit Round Robin), the current available simulation-output data is reporting metrics of simulations whose every node implements only FIFO queues. Regarding our experimental setup, the training was done on an NVIDIA GeForce RTX 2080 Ti in under 20 hours. The code is available on Github[1].

[1]https://github.com/ITU-AI-ML-in-5G-Challenge/ITU-ML5G-PS-001-SOFGNN-Graph-Neural-Networking-Challenge



(a) Training **(b)** Validation **(c)** Test

Fig. 4 – Sample networks

Table 2 – Hyper-parameters

	GNN_1	GNN_2
Link Hidden State Size	256	128
Path Hidden State Size	256	128
Readout Layer Size	256	128
Message Passing Rounds	4	4
Learning Rate	0.001	0.001
Optimizer	Adam	Adam
Loss	MAPE	MAPE

6.2 Hyper-parameter tuning

After some hyper-parameter searches, we observed that a learning rate of $lr = 10^{-2}$ seems to be too large as step, whereas $lr = 10^{-4}$'s training takes longer to converge with no apparent improvement in MAPE over $lr = 10^{-3}$. Regard- ing the number of message passes, we did not observe any improvement in the performance of the model when we in- crease it. As this hyper-parameter has the most drastic ef- fect on the training time of the model, we see no incentive to set the value beyond 4. Regarding the link hidden state size, we observed some improvement in the validation re- sults up to a size of 256 units, beyond which improvement is harder to discern. Likewise, we observe no apparent ben- efit in increasing this value beyond 256, as the validation performance noticeably degrades past this point. Based on these observations, Table 2 presents in detail the hyper- parameters we used while Table 3 presents the architecture of our proposed GNN ensemble model.

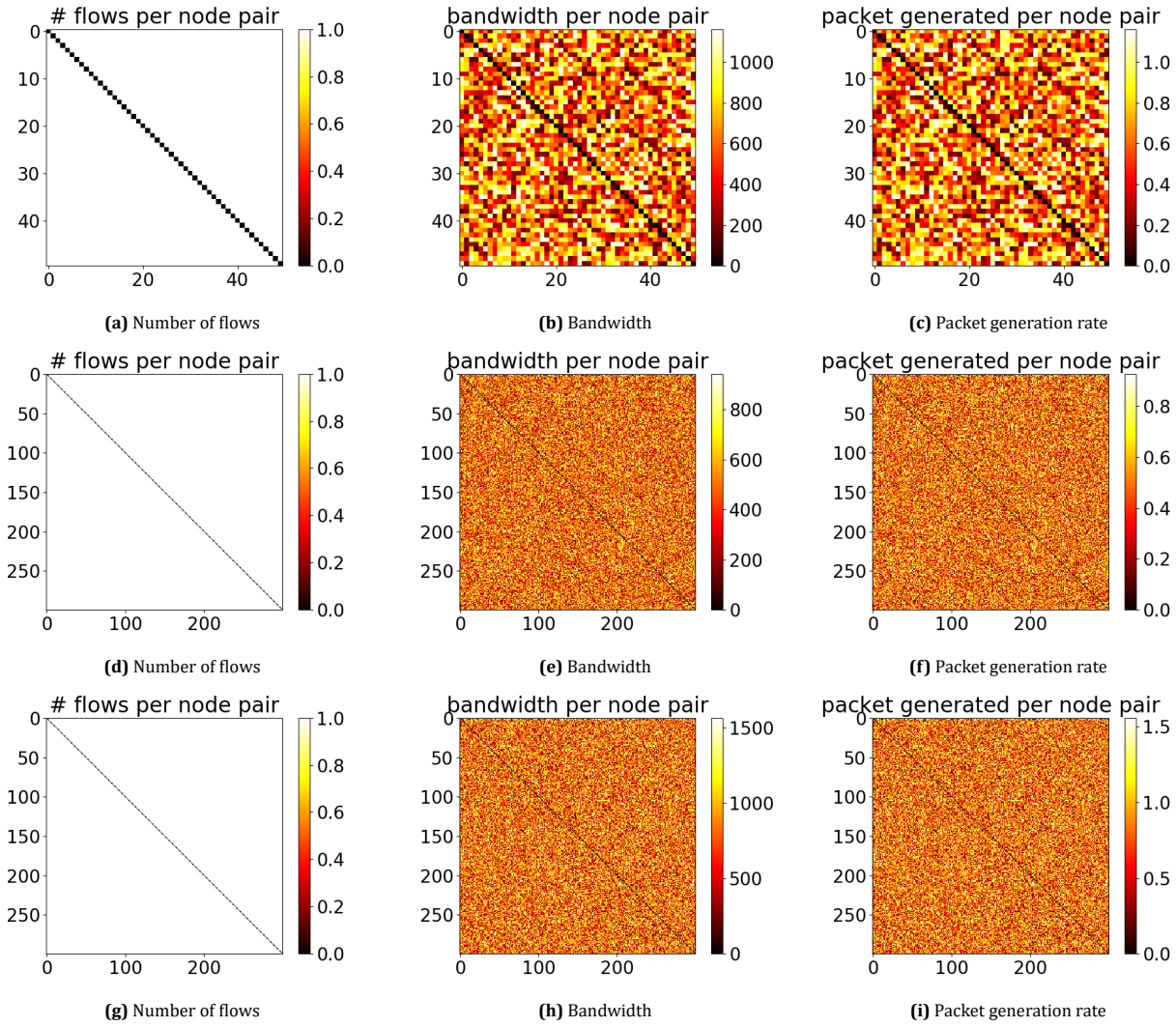
6.3 Link bandwidth vs link load

We conducted some experiments to evaluate and validate the importance of our feature design. We compare the results of training our GNN model using the "capacity", or link bandwidth feature elaborated in the original RouteNet configuration in one hand and using our designed feature "Link Load" that we have defined in Section 5.2 on the other hand. Fig. 6 shows the training and validation curves for the two GNN models with all hyper-parameters identical (the ones of GNN_1 presented in Table 2). We observe that the swapping of "link bandwidth" feature to "link load" feature results in a significant increase in the accuracy of the model on the validation set. Likewise, a significant improvement in the stability of the training relative to before the change was made is observed. The generalization ability of the model with the proposed feature is therefore improved as shown by the narrowing of the train-validation accuracy gap.

In Table 4, we present the overall MAPE of the two models on the validation and test set. We observe that, simply swapping these two features results in a drop of the final MAPE on both validation and test sets from around 50% down to around 2%.

6.4 Queue occupancy vs. direct delay

In this section, we investigate the effect the form of the model's output has on it's accuracy, in order to verify claims that a model trained to predict a variable whose distribution varies much less across training, validation and test sets and different topology sizes will generalize better as the size of the topology shifts. Table 6 shows the results of both the direct delay prediction model and the queue occupancy prediction model on the validation and test sets.



(a) Number of flows **(b)** Bandwidth **(c)** Packet generation rate

(d) Number of flows **(e)** Bandwidth **(f)** Packet generation rate

(g) Number of flows **(h)** Bandwidth **(i)** Packet generation rate

Fig. 5 – Datasets visualization: Training set from (a) to (c) , validation set from (d) to (f) and test set from (g) to (i). The values of the number of flows, bandwidth and packet generation rate are given per node pair.

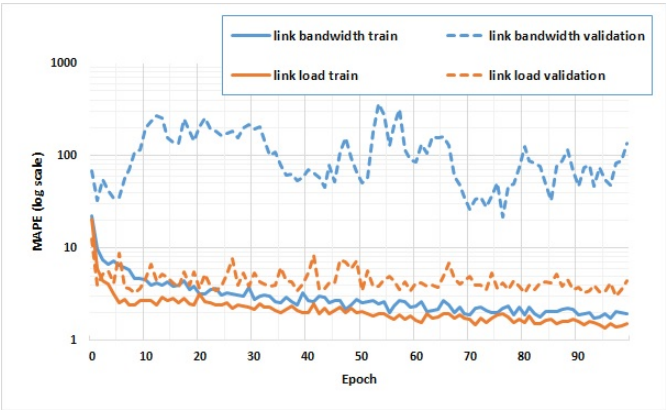


Fig. 6 – Comparison of training and validation curves for model with link bandwidth as a feature vs link load

Both models are identical save for their output type, using the same hyper-parameters as the previously-mentioned GNN_1. We can see that the model utilizing the queue occupancy and per-flow delay estimation configuration clearly outperforms the direct delay model, as can be understood by the value of the MAPE going from above 200 down to under 2, demonstrating it's superior ability to generalize to topologies of increasing size.

6.5 Stability analysis

As explained in Section 5, our final configuration is an ensemble of two GNN-based models with inputs and parameters defined respectively in tables 1 and 3. From the results of Section 6.3 and Section 6.4, we have the confirmation that using our proposed feature design and output de-

Table 3 – Graph neural networks architectures

Layer	Type	Activation	GNN_1 Number of Units	GNN_2 Number of Units
Link Embedding				
1	Fully conected	RELU	256	128
2	Fully conected	SELU	256	128
Path Embedding				
1	Fully conected	RELU	256	128
2	Fully conected	SELU	256	128
Link Update				
1	GRU	-	256	128
Path Update				
1	GRU	-	256	128
Readout				
1	Fully conected	RELU	32	128
2	Fully conected	RELU	32	128
3	Fully conected	None	1	1

Table 4 – Link bandwidth vs link load

	Baseline Model	Our model v1
Flow features	Traffic Packets EqLambda	Traffic Packets EqLambda
Link features	Link Bandwidth	Link Load
	MAPE(%)	MAPE(%)
Validation set	57.90	1.94
Test set	46.82	1.86

sign significantly improves the results. We have conducted additional experiments to evaluate the stability of the proposed model. We retrained the model six times and evaluated its performance for each run on the test and validation datasets. Table 5 shows the results we obtained over the six runs, and the last column presents the mean and standard deviation on both the validation and test sets for each run. We observe that taking the ensemble of both models helped in improving the performance. We moved from two models of around 1.5% of MAPE on average to one ensemble model with around 1.3% of MAPE on average, on both validation and test sets. The similarity of results on both datasets was expected as they both contain samples following similar distribution. Moreover, the ensemble model yields a much more stable MAPE with a standard deviation of about 0.01 in the validation set and 0.055 in the test set.

6.6 Robustness analysis

To further investigate how scalable the proposed model is, we evaluated how well the model will perform for samples of networks' sizes in different ranges. Indeed, we considered five different ranges as shown in Fig. 7. We can see that the solution is consistent regardless of the size of the network with a median less than 1.4% for all ranges. Despite the fact that the variability does not show a trivial pattern as the size of the network increases, we still notice that we have two levels of median on this figure. From 55 to 150, there is an approximate median of about 1.15% and from 160 to 300 an approximate median of about 1.4% which is still a very low MAPE. This shows that the proposed solution is highly scalable. However, given the small increase in this median value over the last interval [220-300], it would be interesting to analyse at what size the performance of the proposed solution starts to degrade considerably. This analysis is left for future work.

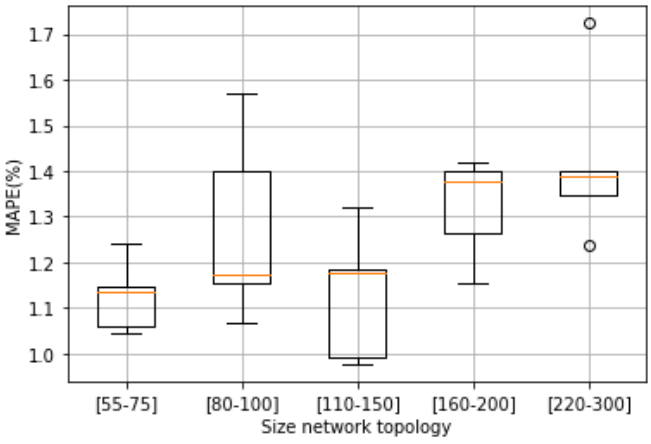


Fig. 7 – MAPE of the ensemble model per range on the test set

Table 5 – MAPE(%) of the proposed model on six independent runs

	Run 1		Run 2		Run 3		Run 4		Run 5		Run 6		Mean± std	
	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test	Val	Test
GNN_1	1.55	1.58	1.62	1.44	1.36	1.32	1.50	1.44	1.75	1.58	1.52	1.40	1.55±0.12	1.46±0.10
GNN_2	1.55	1.54	1.55	1.54	1.53	1.63	1.49	1.46	1.41	1.46	1.38	1.30	1.48±0.07	1.48±0.11
Ensemble	1.36	1.33	1.35	1.24	1.36	1.39	1.33	1.32	1.38	1.34	1.36	1.24	1.32±0.01	1.31±0.05

Table 6 – Direct delay prediction vs. queue occupancy prediction

	Baseline Model	Our model v2
Prediction	Direct Delay	Queue Occupancy
	MAPE(%)	MAPE(%)
Validation set	292.92	1.75
Test set	248.28	1.58

7. CONCLUSION

We have shown that we can improve the performance of the RouteNet model on the delay prediction task through a few small modifications to the model's input variables and output. The first change is the replacement of the "capacity", or link bandwidth, link feature present in the default configuration of Routenet, with a hand-designed feature we call "link load", consisting of the total amount of traffic going through the link as a function of the link's capacity. The second change concerns the output of the model. Instead of having the model directly output the per-path delay predictions, we have the model output per-link queue occupancy predictions and infer the per-path delay from these using a simple transformation, allowing us to avoid the "Out of Distribution" problem encountered by the previous configuration when attempting to generalize to larger topologies, as the distribution of queue occupancy varies less with topology size.

Results show that implementing both of these changes had a significant effect on the accuracy of the model, lowering the MAPE on the validation and test sets from the 300 range down to less than 2. Transitioning from a direct delay prediction model to one where the formula presented in Section 5.3 leads to a reduction in the MAPE on the validation and test sets from 300 down to around 50, while the replacement of the "capacity" feature with "link load" further reduces the MAPE on the training and validation set, bringing it to under 2.

While the results presented in this work show that the solution scales relatively well as topology size increases, it would be of interest to find out exactly to what extent this holds by going beyond 300 nodes to test on even larger topology networks. Verifying if the current solution could also be applied to predict other types of common KPIs such as jitter and packet loss may be another interesting research direction to pursue.

ACKNOWLEDGEMENT

This work was supported by several joint Mitacs-Ciena (Large-scale optimization for optical and fiber networks & Self-Organized Fabric - SOF projects) internships, as well as from the ministère de l'Économie et de l'Innovation for CRIM.

REFERENCES

[1] D. Kreutz, F.MV. Ramos, P.E. Verissimo, C.E. Rothenberg, S. Azodolmolky, and S. Uhlig. "Software-defined networking: A comprehensive survey". In: *Proceedings of the IEEE* 103.1 (2014), pp. 14–76.

[2] J.E. Preciado-Velasco, J.D. Gonzalez-Franco, C.E. Anias-Calderon, J.I. Nieto-Hipolito, and R. Rivera-Rodriguez. "5G/B5G Service Classification Using Supervised Learning". In: *Applied Sciences* 11 (2021), p. 4942.

[3] M. Xie, F. Michelinakis, T. Dreibholz, J.S. Pujol-Roig, S. Malacarne, S. Majumdar, W.Y. Poe, and A.M. Elmokashfi. "An Exposed Closed-Loop Model for Customer-Driven Service Assurance Automation". In: *Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. 2021, pp. 419–424.

[4] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio. "RouteNet: Leveraging Graph Neural Networks for network modeling and optimization in SDN". In: *IEEE Journal on Selected Areas in Communications (JSAC)* 38.10 (Oct. 2020), pp. 2260–2270.

[5] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, and G. Monfardini. "The graph neural network model". In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80.

[6] F. Geyer and S. Bondorf. "Graph-Based Deep Learning for Fast and Tight Network Calculus Analyses". In: *IEEE Transactions on Network Science and Engineering* 8.1 (2021), pp. 75–88.

[7] K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros, and A. Cabellos-Aparicio. "Unveiling the Potential of Graph Neural Networks for Network Modeling and Optimization in SDN". In: *ACM Symposium on SDN Research (SOSR)*. San Jose, CA, USA, 2019, pp. 140–151.

[8] J. Suárez-Varela et al. "The graph neural networking challenge: a worldwide competition for education in AI/ML for networks". In: *ACM SIGCOMM Computer Communication Review* 51.3 (2021), pp. 9–16.

[9] G. Bernárdez, J. Suárez-Varela, A. López, B. Wu, S. Xiao, X. Cheng, P. Barlet-Ros, and A. Cabellos-Aparicio. "Is Machine Learning Ready for Traffic Engineering Optimization?" In: *IEEE 29th International Conference on Network Protocols (ICNP)*. 2021, pp. 1–11.

[10] N. Vesselinova, R. Steinert, D.F. Perez-Ramirez, and M. Boman. "Learning Combinatorial Optimization on Graphs: A Survey With Applications to Networking". In: *IEEE Access* 8 (2020), pp. 120388–120416.

[11] DataNet API Documentation. `https://github.com/BNN-UPC/datanetAPI/tree/challenge2021`.

[12] A. Varga and R. Hornig. "An overview of the OMNeT++ simulation environment". In: *International conference on Simulation tools and techniques for communications, networks and systems & workshops*. 2008, pp. 1–10.

[13] K. Xu, M. Zhang, J. Li, S.S. Du, K.-I. Kawarabayashi, and S. Jegelka. "How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks". In: *International Conference on Learning Representations (ICLR)*. 2021, pp. 1–52.

[14] J. Suárez-Varela et al. *Dataset Graph Neural Networking Challenge 2021 Creating a Scalable Network Digital Twin*. URL: `https://bnn.upc.edu/challenge/gnnet2021/dataset/`. (accessed: 01.08.2021).

AUTHORS

Junior Momo Ziazet received his M.Sc. degree in telecommunications engineering from the National Polytechnic School of the University of Douala, Cameroon in 2017. In 2020, he received another M.Sc. in industrial mathematics from the African Institute for Mathematical Sciences. He is currently pursuing a Ph.D. in computer science at Concordia University, Canada. His main research interests focus on large-scale optimization and the application of machine learning and deep learning approaches to communications and network systems.

Charles Boudreau received a B. Sc. and M. Sc. in computer science from Concordia University, Canada in 2018 and 2021, respectively, where he is currently pursuing a PhD. His research interests include graph neural networks, and deep learning applications towards optimization of service function chains in cloud networks.

Brigitte Jaumard is a professor in the Computer Science and Software Engineering (CSE) Department at Concordia University. Her research focuses on mathematical modeling and algorithm design (large-scale optimization and machine learning) for problems arising in communication, transportation and logistics networks. She is also a senior advisor for the Montreal Ericsson GAIA (Global Artificial Intelligence Accelerator) research center and the chief scientist of CRIM. Brigitte Jaumard was ranked among the top 2% of scientists in her field of research according to a 2021 study based on research citations. She was awarded several research chairs (Canada Research Chair and Concordia Research Chair, both Tier I during the years 2000-2019). B. Jaumard has published over 300 papers in international journals in operations research and in telecommunications.

Huy Duong Duong, Quang Huy, Ph.D., is a postdoctoral researcher with CRIM - Computer Research Institute of Montreal (Ph.D. in computer science, Concordia University, Canada, 2020; Bachelor of Information Technology, Hanoi University of Science and Technology, Viet Nam, 2014). His interests include operations research, large-scale optimization, applied AI and ML in networking. Huy is currently involved in the research coordination of the Self-Optimizing Fabric (SOF) project of Ciena, i.e., a project addressing the complexity of distributing intelligence across disparate intelligent systems collaborating toward mutual tasks while maintaining a separation of concerns.