A Practical Network Digital Twin for IGP Weight Optimization

Mohamed Zalat
Systems and Computer Engineering
Carleton University
Ottawa, Canada
mohamedzalat@sce.carleton.ca

Maede Davoudzade
Systems and Computer Engineering
Carleton University
Ottawa, Canada
maededavoudzade@sce.carleton.ca

Chris Barber
Blue Planet
Ciena
Ottawa, Canada
cbarber@ciena.com

David Krauss
Advanced Development
Visionary Technologies, Inc.
Baltimore, USA
dkrauss@visionaryinc.net

Babak Esfandiari
Systems and Computer Engineering
Carleton University
Ottawa, Canada
babak@sce.carleton.ca

Thomas Kunz
Systems and Computer Engineering
Carleton University
Ottawa, Canada
tkunz@sce.carleton.ca

***Abstract*—Current research on Network Digital Twins (NDT) focuses on building models that predict the behavior of the network on simulations rather than building a complete NDT that automatically monitors a physical network and optimizes it according to its current state and traffic flows. In this paper, we implement an online learning NDT for Interior Gateway Protocol (IGP) weight optimization that monitors a physical network to find IGP weight configurations based on a selected objective, such as minimizing average traffic delay, and automatically applies them to the network. We addressed the practical issues of implementing a NDT on a real network, such as detecting and dealing with an improperly trained digital model of the network. Our results indicate that our NDT can effectively optimize a physical testbed and that fine-tuning its digital model on high traffic scenarios improves the performance of the NDT on a small network.**

***Index Terms*—Network Digital Twins (NDT), Traffic Engineering, IGP Weight Optimization, RouteNet-Fermi, Fine-tuning**

I. Introduction

A. Motivation

Digital Twins (DT) of communication networks, known as Network Digital Twins (NDT), are an idea that has been theorized in recent years with a current work-in-progress Internet Research Task Force (IRTF) draft to standardize them [1]. Research efforts on NDTs have mostly focused on developing models that predict the behavior of communication networks rather than creating a NDT that monitors a physical network and optimizes it in real-time [2]–[7]. Creating such a NDT poses many challenges; for instance, how to continuously monitor the topology and interface information from the network and apply configurations automatically. Furthermore, a NDT that relies on Machine Learning (ML) models can sometimes inaccurately predict the behavior of the network resulting in undesired effects that can negatively impact the physical network. At the time of writing, no published study has implemented a real-time NDT for Interior Gateway Protocol (IGP) weight optimization on a physical network.

In the field of IGP weight optimization, most methods focus on minimizing the maximum link utilization in the network [8]–[11] or the energy consumption of the network [12]. Only one method considers delay in the Service Layer Agreements (SLA) of the traffic flows in the network when optimizing IGP weights [11]. Those algorithms require knowledge of the topology and flows of the network a priori and do not automatically optimize the network when it changes. Using a NDT, we can monitor the network for changes and continuously optimize its IGP weights for the current traffic flows and its current topology. Moreover, we can use our NDT to optimize the IGP metrics for the delay and loss of traffic flows directly, instead of just link utilizations. This allows us to meet SLAs without using more complicated methods such as Multiprotocol Label Switching (MPLS) and Segment Routing (SR).

B. Contributions

In [13], we proposed a NDT that uses RouteNet-Fermi, a Graph Neural Network (GNN) based model proposed by Ferriol-Galmés *et al.* [4], and a genetic algorithm to optimize IGP weights. We show how the NDT can make poor decisions that negatively impact the network in Section V, which may result in the operator losing trust in the NDT. We address this issue by extending [13] with an online learning component and testing our NDT on a physical network to observe how well our NDT optimizes the IGP weights of the network under different scenarios. Our study is similar to the work of Ferriol-Galmés *et al.* [7] where they use a physical network to collect a data set to train and test the accuracy of their model's predictions.

Lastly, we perform an online learning experiment to quantify the number of training samples needed from our physical network to re-calibrate a pre-trained RouteNet-Fermi model

We acknowledge the support of Ciena and the Natural Sciences and Engineering Research Council of Canada (NSERC).

978-3-903176-66-9 ©2024 IFIP

used in our NDT to confidently optimize the IGP weights of our network. In the next section, we briefly describe the IGP weight optimization problem and introduce some of the methods in the literature. We also discuss current research efforts in creating DTs of communication networks.

II. State of The Art

To understand where we place our contribution, we discuss the current literature in the field of IGP weight optimization and the field of NDTs in this section. We divide this section into two parts: Section II-A defines IGP weight optimization and introduces the current literature on IGP weight optimization approaches, and Section II-B discusses what a NDT is and current research efforts in developing NDTs.

A. IGP Weight Optimization

Open Shortest Path First (OSPF) and Intermediate System to Intermediate System (IS-IS) are two link-state routing protocols that rely on a link metric in the network known as the IGP weight. The IGP weight of a link denotes the cost of taking the link. Link-state routing protocols route traffic to their destination based on the minimum-cost path to their destination. IGP weight optimization is the process of setting the IGP metric of each link in the network to optimize for some metric of the network, such as maximum link utilization, average End-to-End (E2E), loss, etc.

As mentioned in Section I, the majority of research focused on proposing methods to optimize IGP weights for maximum link utilization [8]–[11] and energy consumption [12]. There are two main approaches for IGP weight optimization: local search and heuristic approaches, and a machine learning IGP weight optimization approach. While local search approaches can sometimes be slower than a machine learning approach, they are simple to implement and provide better solutions [9].

While current techniques optimize IGP weight configurations in a short period, none of them consider any optimization target beyond link utilization. They also do not monitor for changes in the network such as traffic flows and topology. A NDT implementation of IGP weight optimization allows us to monitor the network and automatically apply the IGP weight configuration. This lets us automatically adapt the network for different scenarios as they arise and reduces human involvement. In the next subsection, we introduce NDTs and some of the literature on NDT applications to identify gaps in current NDT applications.

B. Network Digital Twins (NDT)

As mentioned in Section I, a NDT is a DT of a communication network [2]. It encompasses the virtual model of the network, and all modules involved in the bidirectional data flow between the virtual model and the physical model. The virtual model in a NDT provides a framework for testing different traffic flow and network configuration scenarios without affecting the physical network. Statistics and metrics such as the average delay, loss, and jitter per flow can be obtained from the virtual model for a given scenario. The optimizer component of the NDT collects data from the physical network, explores different network configurations using the virtual model, and finally applies a configuration to the network once a better configuration is found. In this section, we discuss some of the literature on the applications of NDTs and summarize the different components used by each paper in a table.

1) Minimizing 5G Network Energy Consumption: Dong *et al.* [14] used a NDT to minimize energy consumption in 5G networks by optimizing resource allocation, offloading policies, and user association subject to Quality of Service requirements. They use a mathematical NDT model to train a deep neural network to determine the optimal user association. Dai *et al.* [15] also used a NDT to monitor the parameters of the network and optimize offloading computational tasks to base stations in an *Industrial Internet of Things (IIoT)* to minimize energy consumption using deep reinforcement learning. Both NDTs were applied to a numerical simulation instead of a physical network.

2) Network Slicing Management: Wang *et al.* [16] use a NDT for *network slicing management*: a method of creating multiple virtual logical networks over a common physical network [17]. Their NDT model is a GNN that allows them to predict the end-to-end latency of network slices under different topologies of the physical network. They use existing datasets collected from different topologies to train and evaluate the accuracy of their model.

3) Meeting Traffic SLAs: Ferriol-Galmés *et al.* [7] use a NDT model based on GNNs and *Recurrent Neural Networks (RNN)* called *TwinNet* to optimize the routes that individual traffic flows take to meet SLAs. They find the route that each traffic flow should take by iteratively modifying the route of each traffic flow, where each flow follows a fixed route that can be configured using MPLS (i.e. it is unaware of the IGP weights of the topology), until all SLAs are met. They also employ TwinNet to find queueing and scheduling policies that meet their SLAs. The objective of their work is to showcase how TwinNet can be used in a NDT for different traffic engineering applications. Hence, we will be using the latest version of their NDT model in our NDT, called *RouteNet-Fermi*, as it was shown to be useful in applications where we need to optimize for QoS metrics such as delay and loss.

In Table I, we summarize some of the main papers that employ NDTs and classify their virtual model and whether they had a physical network as their twin. We can see from Table I that none of the approaches were directly retrieving data and applying configurations to a physical test-bed in real-time.

In this section, we saw how current IGP weight optimization approaches do not provide a generic objective, where the operator can specify what to optimize the IGP weights for. We also saw that there are no reports of NDTs that monitor a real network. To address this, we provide a generic IGP weight optimization approach using a NDT where the operator defines the objective to optimize for. We also perform our experiments on a physical testbed where our NDT collects

TABLE I: A summary of NDT contributions

Paper	Digital model	Trained Online?	Physical Twin?
Dong *et al.* [14]	Mathematical	N/A	✗
Dai *et al.* [15]	Mathematical	N/A	✗
Wang *et al.* [16]	GNN	✗	✗
Ferriol-Galmés *et al.* [7]	GNN + RNN	✗	✗

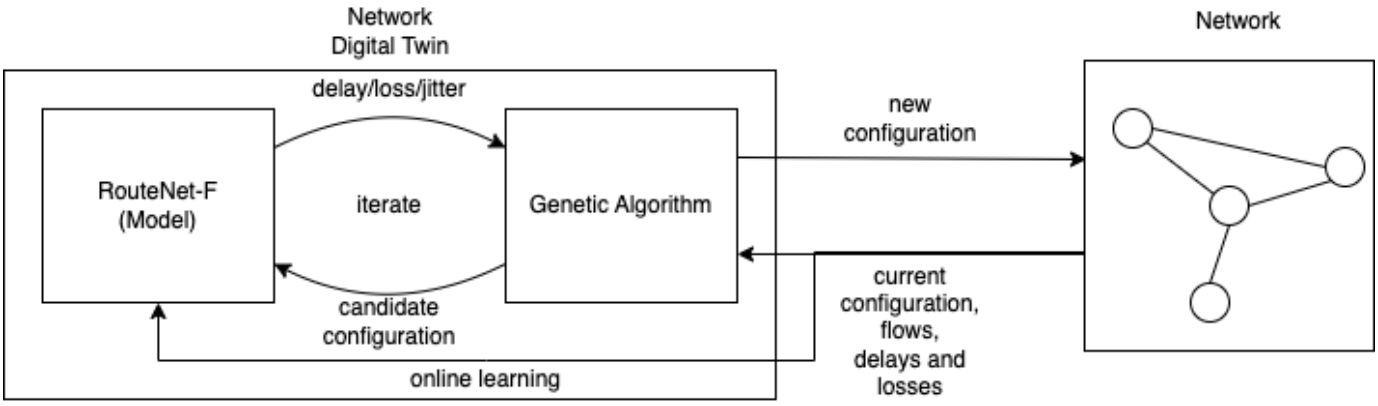


Fig. 1: Online learning NDT for Quality of Service (QoS) based IGP weight optimization.

data from the network in real-time and configurations are automatically applied to the network.

III. Background

In this section, we introduce some of the algorithms and models used to build our IGP weight optimization NDT. We start by introducing the ML model used in our NDT, RouteNet-Fermi [4].

A. RouteNet-Fermi

Fig. 2: Inputs and outputs of RouteNet-F. [4]

RouteNet-Fermi, also known as RouteNet-F, is a GNN model of communication networks proposed by the same authors as TwinNet, Ferriol-Galmés *et al.* [4]. It uses the topology of the network (including link bandwidths, queuing policy of each interface, and the number of queues at each interface and their respective sizes), the traffic flows between each source-destination node (described by the average bandwidth demand, and the packet size and time distributions), and the route each traffic flow takes as input. The resulting output of RouteNet-F is the predicted delay, loss, and jitter of each traffic flow. Their results show that RouteNet-F can generalize and accurately predict QoS metrics on unseen topologies and traffic flows. It serves as a ML-based network model that is capable of predicting QoS metrics much faster than tools such as *OMNeT++* which makes it a good fit for the virtual model of our NDT.

In the following subsection, we introduce online learning, the method used in our NDT to make sure that the IGP weights applied by our NDT do not impact the network negatively.

B. Online Learning

Online learning is a ML method where a ML model is trained using one data point at a time from a sequence of data [18]. For our NDT, we assume that we have full feedback on the data we receive and hence we focus on supervised online learning. This form of online learning involves a sequence of data where full feedback on its predictions is provided, i.e. the ground truth for its input is provided.

Online Gradient Descent (OGD) is one method of supervised online learning where a learning step is used to update the weights of a neural network after each prediction [19]. To ensure that the IGP weights being applied by our NDT are based on valid predictions, we use OGD to train on the measurements obtained from the physical network when RouteNet-F's predictions are not correlated to the physical network. This step is important because we do not want our NDT to apply IGP weight configurations that can worsen the overall QoS of the traffic flows in the physical network.

In the next section, we bring all the components discussed in this section together to describe the architecture of our online learning NDT for IGP weight optimization.

IV. Online Learning NDT for IGP weight optimization

A NDT allows us to test IGP weight configurations without affecting the traffic of the real network. Using the data collected from the physical network, we can use a NDT to

search the IGP weight configuration space. The NDT's model provides us with the predicted QoS metrics of each traffic flow, allowing us to fine-tune our IGP weights in the NDT until we find a configuration that meets our requirements. Once found, we can apply this configuration to the network while still monitoring the traffic flows and the network in case of any changes to the network or its traffic. Hence, the ideal NDT for QoS-based IGP weight optimization should have three essential components: (i) monitoring of the network to detect changes in the network and its traffic, (ii) quick and accurate predictions of the traffic flows' QoS metrics for any given topology, and (iii) the ability to automatically modify the IGP weights of the network. We use RouteNet-F as the model within our NDT to address the second component. The first and third components are a matter of having modules that interface with the devices in the network to collect information, search the IGP weight configuration space, and apply configurations to the devices.

In Figure 1, we present the architecture of our online learning NDT for IGP weight optimization. Our NDT continuously polls the network for information on its current topology including the bandwidths and queue sizes of the interfaces in the network. It also collects information on the traffic flows going through the network and their respective delays and losses. Then, it compares the delay and loss measurements from the network to the predictions of RouteNet-F to decide whether it should train RouteNet-F or optimize the IGP weights of the physical network. If the measurements from the network are not positively correlated with the predictions of RouteNet-F, with a statistical significance α, it continuously trains RouteNet-F on new samples collected from the network until it is positively correlated. Each new sample is collected when a change in the network occurs (including flow changes). Once the measurements from the network and the predictions of RouteNet-F are sufficiently correlated, it executes a genetic algorithm to try different IGP weight configurations on RouteNet-F to search for a better configuration for the current scenario. Once a better IGP weight configuration is found, it is applied to the physical network.

A. Genetic Algorithm

We use the same algorithm described in our previous publication [13]. Each candidate IGP configuration generated is ranked based on a fitness function that ranks a configuration based on the predicted delays and losses of the traffic flows from RouteNet-F. For instance, we used the negative average traffic delay as the return value of our fitness function in one of our experiments. The highest scoring configuration is then mutated to create the next generation until no improvements are found within 4 generations. Once it terminates, the NDT applies the new IGP weight configuration to the physical network by connecting to each router and applying the respective IGP weights to its interfaces. Our online learning NDT polls the network for its current topology and traffic flows at 5-minute intervals and executes the genetic algorithm to adjust the IGP weights of the network accordingly. For our digital model, M, we use RouteNet-F. In the next subsection, we introduce the algorithm of our online learning NDT, which uses the genetic algorithm to optimize the IGP weight configuration for the physical network.

B. Online Learning

To optimize the IGP weights of the network for traffic QoS, it is sufficient for the QoS predictions, such as traffic delays, of RouteNet-F to correlate with the actual QoS metrics in the network. In other words, when the QoS metrics of a traffic flow decreased due to a change in the network, then the predictions from RouteNet-F should also decrease even if they do not match in absolute value. In cases where our delay predictions from RouteNet-F are not correlated with the values in the real network, we augment RouteNet-F's training data with obtained measurements from the network until there is a sufficient correlation between the prediction of RouteNet-F and the network before optimizing IGP weights. To determine when to train RouteNet-F, we set a threshold correlation (specifically, the Spearman correlation coefficient [20]) value, below which we train RouteNet-F on samples from the physical network until our correlation exceeds this threshold. Once the correlation coefficient between the predicted and measured values exceeds a preset threshold, we begin optimizing the IGP weights of the network until it drops below the threshold again.

We assume that we have at least 3 flows in our network so we can perform a one-tailed Student t-test based on $f-2$ degrees of freedom, where f is the number of flows in the network. Suppose t is above the 95^{th} percentile, $t_{\alpha=0.05}$, for its corresponding $f-2$ degrees of freedom in Student's t-distribution. In that case, we can conclude a significant correlation exists between the delays predicted by RouteNet-F and the delays measured from the network, with a 0.05% statistical significance. We perform an experiment to determine the optimal value for α on our physical network in Section VI. In cases where there are 2 flows or fewer, we assume that the RouteNet-F model predictions are accurate and our genetic algorithm for IGP weight optimization is executed.

In Algorithm 1, we describe our online learning NDT. As described, our algorithm executes periodically every c minutes or on a network change (including traffic changes). Our NDT continuously polls the network by logging in to each router and collecting information on each interface including their status, queues, bandwidths, and IGP weights. When a network change is detected (or c minutes have passed), our NDT checks if the calculated t is below the preset α threshold of its corresponding Student's t-distribution using the measurements from the network and trains RouteNet-F if t is below the threshold. Once the calculated t exceeds the t_α threshold, we optimize the IGP weights of the network instead of learning from the network. The goal of this is to ensure that our model is sufficiently accurate before we optimize the network.

V. Preliminary Experiment

To test how well a pre-trained NDT optimizes the IGP weights of the network when RouteNet-F poorly predicts the

Algorithm 1: Online Learning NDT

Input: The physical network, *N*,
the digital model, *M*,
the t-test correlation error threshold, α,
the maximum update interval in minutes, c,
the fitness function, *fitnessFunction(config, flows, M)*.

repeat *every c minutes or on network change*
  currentTopology ← retrieve topology and features of *N*
  currentFlows ← retrieve estimated distribution of each flow in *N*
  actual ← Retrieve average delay, jitter and loss per flow from *N*
  predicted ← predict($M, currentTopology, currentFlows$)
  r_s ← spearmanCorrelation($predicted, actual$)
  t ← calculateT($r_s, |currentFlows|$)
  if $t < t_\alpha$ **then**
    train($M, currentTopology, currentFlows, actual$)
    //Retrain *M* with the new configuration/flows and the actual delays, jitters and losses from *N*
  else
    geneticAlgorithm($N, M, fitnessFunction$)
    //Refer to Section IV-A
  end
end

QoS metrics, we plotted a histogram of the link utilizations of the 128-fat-tree dataset used to train RouteNet-F. We saw that only a few cases have a link utilization above 70% in RouteNet-F's training set. To have many links with high utilization, we used a 6-node OMNeT++ topology with constant bit-rate traffic flows between each node pair that demand bandwidths between $18kb/s$ and $25kb/s$, with all links having a capacity of $100kb/s$. The queue sizes were set to $32kb$ and were using First-In First-Out (FIFO) queueing.

We then used this OMNeT++ network as the "physical" twin of our pre-trained NDT, where the fitness function was set to return the negative global average delay. As we can see from our results in Figure 3, the IGP weight configuration recommended by our NDT increased the average delay of our OMNeT++ network instead of decreasing it due to the inaccurate predictions of the pre-trained RouteNet-F model. When the NDT operates on a physical network, this can cause service disruptions and negatively impact the network traffic. While unseen link utilizations are one example of cases where RouteNet-F's predictions can be inaccurate, there may be other cases such as unseen traffic size and time distributions. In Section VI-B, we address those issues by constantly training the RouteNet-F model on the physical network.

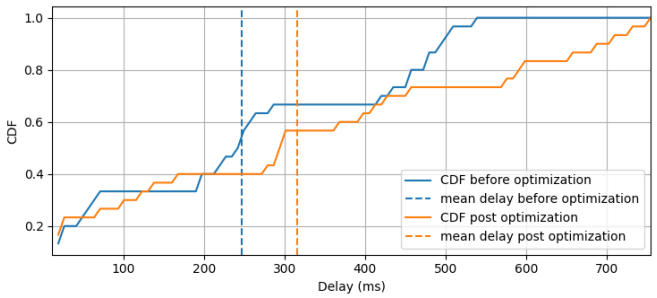


Fig. 3: Results of the NDT when optimizing a network with unseen link utilizations.

VI. Results and Discussions

For our experiments, we used a physical testbed provided by Ciena [21] with 7 routers and an *IXIA* traffic generator where we can specify the traffic flowing through the network (refer to Figure 4a, the numbered nodes are routers and the links between them are L3 links). Our network consisted of Cisco 7200 routers, where each interface (link) has a bandwidth of 1000Mbps and is configured with a queue depth of 1000 frames. All the flows generated in our experiments are constant bit-rate traffic flows with a frame size of 1kByte.

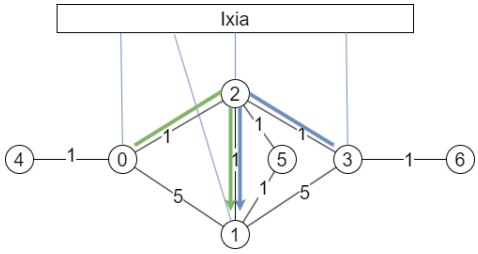
Our NDT monitors the setup by logging in to each router through Secure Shell Protocol (SSH) and collecting queue size, IGP metric, interface status, and bandwidth information for all interfaces to update the topology, which takes about 5 minutes. Hence, if a new router gets added, this needs to be manually added to the NDT routers list so that the NDT can access the router. Our NDT obtains traffic information about the flows and their delays from the IXIA traffic generator. To scale the NDT to a larger production network, traffic flow and topology information should be collected using protocols such as NetFlow [22] and Simple Network Management Protocol (SNMP) to save time. Our NDT also applies the optimal IGP weight configuration by logging in to all the routers and applying the weights to the respective interfaces.

The RouteNet-F model used in our experiments only predicts traffic delays. We use a RouteNet-F model trained on the 128-fat-tree dataset for 15 epochs using the Adam optimizer with a 0.001 learning step.

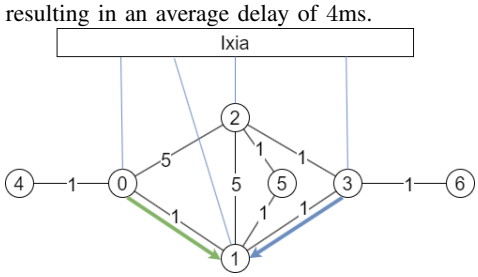
A. Case Study

In the first experiment, we test if the NDT can automatically improve the average traffic delay of the network by changing the IGP weights of the network and to assess the performance in connecting a NDT to a physical network. We measure the time it takes to collect information from the network and the time it takes to apply IGP weights after finding the optimal configuration.

We used two constant bit rate traffic flows of 80Mbps bandwidth each, one going from node 0 to 1 and another from node 3 to 1 with the IGP weight settings in Figure 4a. We also used traffic shaping to limit the bandwidth of all the links in the



(a) In this scenario, two 80Mbps flows share the same path from node 2 to 1 congesting that link, resulting in an average delay of 4ms.



(b) The IGP weights and the new traffic routes after one cycle of the NDT. The result is an improved average traffic delay of 1ms.

Fig. 4: The physical testbed with a traffic flow scenario denoted by the colored arrows. We bottleneck the traffic flows and demo the NDT. The numbers on each link denote its IGP metric.

network to 150Mbps to congest the links. The average delay of both traffic flows is 4ms from IXIA before optimizing the IGP weights for those two flows. After one cycle of our NDT, the IGP weights changed to reroute traffic as in Figure 4b, reducing the average delay of both traffic flows to 1ms. The current implementation of the NDT also changed the weights of the links between nodes 0 and 2, and nodes 2 to 1 to 5 which might congest a future flow going from node 2 to node 0. This congestion should not last long because once a new flow is introduced that is going from node 2 to node 0, the NDT will consider it, and the IGP weights will change back to 1 to minimize the delay of the new flow. To solve this issue we need to change the genetic algorithm to select the best IGP weight configuration with the least number of changes to select configurations that are closer to the current configuration of the network.

To collect data about the interfaces and the routers in the network, our NDT requires to login to all the routers in the network through SSH. Hence, we maintain a database of all the routers in the network and their login information. New routers need to be added manually to this list. This information includes sensitive information about the network and hence we need to store this information securely. Collecting all the information required from the topology on our setup takes approximately 4 minutes. Applying the IGP weights to the network after optimization takes about 2 minutes.

Our NDT successfully improved the average traffic delay of the physical testbed on a preset scenario in this experiment; however, we need to perform more experiments with different scenarios to make a stronger conclusion on its performance on a physical testbed. Moreover, our virtual model was predicting the delays accurately in this scenario which does not tell us how it performs when the predictions diverge from the physical network. In the next section, we test different α thresholds and use a pre-trained RouteNet-F model to determine how many samples we need to collect from the network for each threshold and whether our online learning NDT improves the maximum delay of difficult to predict network traffic scenarios (specifically, high traffic load scenarios) with its recommended IGP weight configurations.

B. Setting the α Threshold for Online Learning

In a second experiment, we use the same pre-trained RouteNet-F model from the previous experiment as our baseline model. Our fitness function selects the IGP weight configuration that minimizes the maximum flow delay in the network. We use $\alpha = 1, 0.03, 0.02, 0.01$ in our online learning algorithm to determine the number of training samples required before optimizing the IGP weights of the network. Since it takes 4 minutes for the NDT to poll our network, we assume that the traffic flow changes at every time-step of the NDT in a live network. Hence, we randomly assign 12 flows between each pair of ports of the IXIA generator with bandwidths between 30Mbps and 50Mbps on each time-step to simulate a live network. We chose this bandwidth range and set the rate of all links to 100Mbps to ensure that the links are as highly utilized as in our preliminary experiment. We also evaluated how well they improved the maximum traffic delay of the network on the first traffic scenario above their α threshold. For our online learning, we used the AdamW optimizer with a 0.0001 learning rate for 10 epochs per sample, and we only update the weights of the final readout layers of RouteNet-F to fine-tune the model. The intuition behind this is to not forget the encoding of the network learnt by RouteNet-F on the old training set. We also repeated this experiment with fine-tuning all the weights of the model (including the hidden layers of the model) and got similar results.

TABLE II: Online Learning NDT results table

α	**Max Delay Reduction (%)**	**Number of Samples Required to Meet Optimization Criteria**
$\alpha = 1$	4.5%	0
$\alpha = 0.03$	15%	1
$\alpha = 0.02$	14%	12
$\alpha = 0.01$	30%	25

Our results in Table II show that fine-tuning the pre-trained NDT, with increasing values of α, helps us optimize the network to minimize maximum delay better in high traffic load scenarios. Even with fine-tuning the RouteNet-F on one sample (at $\alpha = 0.03$), our predictions in high traffic scenarios were better because we perform 10 epochs of fine-tuning per

sample obtained. At $\alpha = 0.01$, we got the best result at a reasonable number of samples from our set of experiments. This means if we collect samples every c minutes, it takes $25 \times c$ minutes to before we can optimize the network. However, in this experiment, we were changing the network traffic flows at each step triggering the algorithm before $c = 10$ minutes have passed. Without fine-tuning the network (at $\alpha = 1$), the maximum delay of the network after applying the IGP weights did not improve by much. Optimizing the network with predictions that are not correlated to its behavior does not improve its state and may even negatively impact it. From our results we can conclude that it is important to fine-tune the NDT's model when our predictions are not correlated with the network's behavior.

VII. Conclusions and Future Work

In this paper, we set up a NDT for IGP weight optimization on a physical test-bed to test its ability to optimize a real network and to learn about what it takes to fully automate the retrieval of data from the network and the optimization of the network. We required interfacing with the router through management interfaces to get queue sizes, bandwidths, link status, IGP weights, and to apply the found optimal weights to the network. We collected traffic information and metrics from IXIA; however, in a real network we would have to obtain this data from routers that support flow monitoring. We used the NDT with a pretrained RouteNet- model in a low intensity traffic scenario to successfully reduce the average traffic delay. We were able to observe a self-adapting network with physical routers and links, where better routes for current traffic flows are found without human intervention.

We also proposed an online-learning NDT to deal with high-traffic load scenarios, where the pre-trained RouteNet-model does not perform well. Our results indicate that using our online-learning NDT with a small $\alpha <= .03$, improves the maximum flow delay in high traffic load scenarios in our physical network and prevents it from making changes that negatively impacts the network. Future experiments include performing the online learning experiment on a larger scale network, and testing our approach with more fitness functions, such as average delay and average packet loss.

References

[1] C. Z. et al., "Network Digital Twin: Concepts and Reference Architecture," Internet Engineering Task Force, Internet-Draft draft-irtf-nmrg-network-digital-twin-arch-05, Mar. 2024, work in progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-nmrg-network-digital-twin-arch/05/

[2] P. A. et al., "Digital twin network: Opportunities and challenges," *CoRR*, vol. abs/2201.01144, 2022. [Online]. Available: https://arxiv.org/abs/2201.01144

[3] M. F. G. et al., "Routenet-erlang: A graph neural network for network performance evaluation," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, London, United Kingdom, May 2-5, 2022*. IEEE, 2022, pp. 2018–2027.

[4] ——, "Routenet-fermi: Network modeling with graph neural networks," *IEEE/ACM Transactions on Networking*, vol. 31, no. 6, pp. 3080–3095, 2023.

[5] S. Xiao, D. He, and Z. Gong, "Deep-q: Traffic-driven qos inference using deep generative network," in *Proceedings of the 2018 Workshop on Network Meets AI & ML, NetAI@SIGCOMM 2018, Budapest, Hungary, August 24, 2018*. ACM, 2018, pp. 67–73.

[6] 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," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2260–2270, 2020.

[7] M. F. G. et al., "Building a digital twin for network optimization using graph neural networks," *Computer Networks*, vol. 217, p. 109329, 2022.

[8] O. Brun and J. Garcia, "Dynamic IGP weight optimization in IP networks," in *IEEE First Symposium on Network/Cloud Computing and Applications, NCCA 2011, Toulouse, France, November 21-23, 2011*. IEEE Computer Society, 2011, pp. 36–43.

[9] K. Rusek, P. Almasan, J. Suárez-Varela, P. Cholda, P. Barlet-Ros, and A. Cabellos-Aparicio, "Fast traffic engineering by gradient descent with learned differentiable routing," in *18th International Conference on Network and Service Management, CNSM 2022, Thessaloniki, Greece, October 31 - Nov. 4, 2022*. IEEE, 2022, pp. 359–363.

[10] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proceedings IEEE INFOCOM 2000, The Conference on Computer Communications, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Reaching the Promised Land of Communications, Tel Aviv, Israel, March 26-30, 2000*. IEEE Computer Society, 2000, pp. 519–528.

[11] A. Nucci, S. Bhattacharyya, N. Taft, and C. Diot, "IGP link weight assignment for operational tier-1 backbones," *IEEE/ACM Transactions on Networking*, vol. 15, no. 4, pp. 789–802, 2007.

[12] F. François, N. Wang, K. Moessner, S. Georgoulas, and K. Xu, "On IGP link weight optimization for joint energy efficiency and load balancing improvement," *Computer Communications*, vol. 50, pp. 130–141, 2014.

[13] M. Zalat, C. Barber, D. Krauss, B. Esfandiari, and T. Kunz, "Network routing optimization using digital twins," in *Companion Proceedings of the 16th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling and the 13th Enterprise Design and Engineering Working Conference: DTE 2023, Vienna, Austria, November 28 - December 01, 2023*, ser. CEUR Workshop Proceedings, vol. 3645. CEUR-WS.org, 2023. [Online]. Available: https://ceur-ws.org/Vol-3645/dte4.pdf

[14] R. Dong, C. She, W. Hardjawana, Y. Li, and B. Vucetic, "Deep learning for hybrid 5g services in mobile edge computing systems: Learn from a digital twin," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4692–4707, 2019. [Online]. Available: https://doi.org/10.1109/TWC.2019.2927312

[15] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Deep reinforcement learning for stochastic computation offloading in digital twin networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4968–4977, 2021. [Online]. Available: https://doi.org/10.1109/TII.2020.3016320

[16] H. Wang, Y. Wu, G. Min, and W. Miao, "A Graph Neural Network-Based Digital Twin for Network Slicing Management," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1367–1376, 2020.

[17] S. Zhang, "An overview of network slicing for 5g," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 111–117, 2019.

[18] S. C. H. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *Neurocomputing*, vol. 459, pp. 249–289, 2021.

[19] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, T. Fawcett and N. Mishra, Eds. AAAI Press, 2003, pp. 928–936.

[20] C. Spearman, "The proof and measurement of association between two things," *The American journal of psychology*, vol. 100, no. 3/4, pp. 441–471, 1987.

[21] C. Corporation, "A networking systems, services, and software company — ciena.com," https://www.ciena.com/.

[22] R. H. et al., "Flow monitoring explained: From packet capture to data analysis with netflow and IPFIX," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.