

Graph-Aware Deep Learning Based Intelligent Routing Strategy

Zirui Zhuang, Jingyu Wang, Qi Qi, Haifeng Sun, and Jianxin Liao

State Key Laboratory of Networking and Switching Technology

Beijing University of Posts and Telecommunications

Beijing, China

{zhuangzirui, wangjingyu, qiqi8266, hfsun, liaojx}@bupt.edu.cn

Abstract—Software defined networking decouples the control plane and data plane, which grants more computing power for routing computations. Traditional routing methods suffer from the complex dynamics in networking, and they are facing issues such as slow convergence and performance decline. Deep learning techniques have shown preliminary results on solving the routing problem, bring more accuracy and precision compared with traditional modeling techniques. However, the deep learning architecture needs to be specially customized to learn the topological relations between switches in an efficient way. Thus, we propose a deep learning based intelligent routing strategy with revised graph-aware neural networks and we design a set of features suitable for network routing. Then we demonstrate the performance of our works by using a real-world topology and the production level software switch. The simulation result shows our work is more accurate and efficient compared to state-of-art routing strategy.

I. INTRODUCTION

With the emerging advancements in Software Defined Networking, the network controllers have more computing power nowadays, which builds the basis of machine learning applications in networking [3]. With OpenFlow [12] and P4 [1] the network achieves better controllability. Furthermore, more aspects of and more accuracy in network state are accessible to the network controllers by recent network measurement evolvments [10]. These progresses in networking lead to the opportunity for more optimal routing decisions.

Network states and dynamics are sophisticated, and situations shift rapidly. There are some factors which are crucial to the strategy of network routing: 1) **The complex dynamics of the networking process.** The network state space is a high-dimensional continuous space which is of great complexity, and different aspects of network state affect the networking dynamics in their own ways. 2) **The performance persistence.** A learning model with long-term stability is needed [14] that it can memorize a mapping between a network state and an optimal future outcome. 3) **The convergence speeds.** When previously unseen situations appear, the routing strategy should be able to learn and converge quickly. Deep learning techniques are ideal to deal with these kinds of obstacles and it is significant to introduce deep learning solutions into network routing.

There has been a recent work on network routing using deep learning techniques [11], which shows pretty good results on simple topologies yet not on complex ones. Empirical

results from machine learning society [5], [6], [13] show that a learning model benefits extensively from graph related information. As a result, we believe the performance of deep learning based routing strategies can be further increased by making the deep learning architecture aware of a network's characteristics.

Therefore, we propose a novel Graph-Aware Deep Learning based intelligent routing strategy (GADL), to unravel the issues mentioned above. In our strategy, we maintain a precise measure of current network state, as well as the current position and destination of forwarding tasks. The network state information is preprocessed into features in a representative form as the input of a revised deep learning architecture. The deep learning architecture consists of a graph kernel and a convolutional neural network. The learning model in this paper is subsequently trained from off-line supervised learning and then tested online. In summary, our main contributions are as follows: 1) We use a centralized deep learning model for routing computation which reduces overhead in networking and model training. 2) We design a feature processing procedure which transforms network state measurements into representative features suitable for a deep learning architecture to learn in a routing scenario. 3) We propose a graph-aware deep learning architecture with a graph kernel which improves efficiency by introducing network topology awareness.

The organization of this paper is as follows. In section II we survey related work. In section III we define the target problem. In section IV we propose a graph-aware deep learning based intelligent routing strategy. In section V experiments and results are presented. And finally, in section VI we conclude the paper and give a discussion.

II. RELATED WORK

A. Routing Methods in Communication Networks

Traditional routing protocols highly rely on shortest-path based algorithms and the pre-defined link weight functions cannot approximate all traffic patterns [4]. Machine learning approaches are also introduced into this area. Sharma et al. [15] use neural networks and decision trees to find next forwarding hop with minimum delay in opportunistic networks. Lee [9] uses deep learning to estimate node degree, and then use the Viterbi algorithm for route selection. The most state-of-art work on network routing is the Deep Belief Architecture

(DBA) from Mao et al. [11]. The DBA is a distributed deep learning based method which takes the packet rates of each node as input and produces next forwarding hop as output.

B. Techniques in Machine Learning

Deep Belief Network [2] is a generative model which uses Gibbs sampling and it is usually considered as a dimensionality reduction tool for sparse features. Deep neural network with feed-forward layers using rectified linear unit (ReLU) [7] achieves a faster and better result for supervised learning tasks with enough data. Convolutional Neural Network (CNN) shares a small set of convolution filters, reducing the cost of the training process [8].

There is a trending interest of bringing graph-awareness into learning models and the works have shown promising results. Duvenaud et al. [6] use hashing and indexing functions to preprocess the input features so that the output feature of each node contains a portion of information from its neighbors. Defferrard et al. [5] use Graph Fourier Transform to bring the computation of convolution into the spectral domain. Niepert et al. [13] use an intuitive method to find partitions of arbitrary graphs with the help of a labeling function.

III. PROBLEM FORMULATION

A communication network consists of hosts, forwarding switches, and communication links between them. A routing task is to find an optimal path between source host's access switch and destination host's access switch in a given network topology. A routing algorithm needs the topological information for finding valid paths, and it needs status information of switches and links for finding optimal paths.

A communication network can be fully specified by a tuple $G(V, E, \{f^v(u)\}, \{f^e(u, v)\})$, where V is a set of vertices denoting switches, E is a set of edges denoting communication links, $\{f^v(u)\}$ is a set of vertex features denoting the status metrics of each switch $u \in V$ and $\{f^e(u, v)\}$ is a set of edge features denoting the status metrics of each link (u, v) where $u, v \in V$ and $(u, v) \in E$.

Routing from source access switch s to destination access switch t is a procedure of selecting a sequence of edges from E and their endpoints from V in consideration of $f^v(u)$ and $f^e(u, v)$, resulting in a subgraph G' where the degree of s and t is 1 and the degree of all other vertices is 2. Let $\langle s, t \rangle$ denotes a routing task from s to t . The direct result of a routing task is a routing path, that is a sequence of forwarding nodes as $\langle s, t \rangle = (u_0, u_1, \dots, u_n)$, where $u_0 = s$ and $u_n = t$. The i -th node in path of routing task $\langle s, t \rangle$ is u_i , noted as $\langle s, t \rangle_i = u_i$.

The routing algorithm selects forwarding nodes one after another iteratively. Let P be the routing procedure. In each iteration, $P(u_n, t, G)$ selects a next hop vertex u_{n+1} connected to current vertex u_n , and then it executes the next iteration $P(u_{n+1}, t, G)$. That is,

$$\langle s, t \rangle_{n+1} = P(\langle s, t \rangle_n, t, G) \quad (1)$$

where $u_0 = s$. As a result, our main interest is to find a procedure P solving the routing path problem defined in (1).

IV. SOLUTION

The system of GADL consists of three phases as shown in Fig. 1. Firstly, the controller in software defined networks keeps track of network state measurements from time to time. Secondly, the switch may request the controller for optimal path upon new traffic flows' arrival. Finally, an intelligent routing model in the controller will calculate an optimal path and then let the controller install it into data plane. We train GADL off-line by engineering history data of the network and test it online when the training is finished.

There are two key factors in GADL. One is feature acquisition and the other is the neural network setup. We are going to illustrate our proposed solution from these two aspects.

A. Feature Acquisition

Let $a^v(u)$ be the attributes of vertex u , $a^e(u, v)$ be the attributes of edge (u, v) , and $c^e(u, v)$ be the full capacity of each link (u, v) . The raw attributes in use are link connectivity, link latency, receive bit rate, receive packet rate, transmit bit rate, and transmit packet rate. We use an adjacency matrix to represent network links. For other attributes, we define the edge features $f^e(u, v)$ as a relative logarithm of free capacity to the full capacity $f_i^e(u, v) = \log(1 + (c^e(u, v) - a_i^e(u, v))/(c_i^e(u, v)))$.

Since network measurements are made on links, the vertex attributes $a^v(u)$ is calculated from the attributes of the edges $a^v(u) = \sum_{v \in \{v | a_i^e(u, v) = 1\}} a_i^e(u, v)$. And a vertex features are defined as a load index relative to its adjacent vertices. $f^v(u) = (a^v(u)) / (\sum_{v \in \{v | (u, v) \in E\}} a^v(v))$

To introduce routing history information into the selection procedure, we strip features of used vertices and edges from tuple G . Let V'_n be the set of used vertices at step n , $V'_n = \{\langle s, t \rangle_i \mid 0 \leq i < n\}$, $E'_n = \{(u, v) \mid u \in V'_n \text{ or } v \in V'_n\}$. In this way, the modified tuple G'_n becomes $G'_n = (V \setminus V'_n, E \setminus E'_n, \{f^v(u)\}, \{f^e(u, v)\})$. Finally, the node selection procedure with fine-processed features is as in

$$\langle s, t \rangle_{n+1} = P(\langle s, t \rangle_n, t, G'_n) \quad (2)$$

B. Graph-Aware Neural Network

We design a graph-aware convolutional structure in which k neighbors are explored for each root vertex and w root vertices are selected in total. When exploring the neighbors, we build a tree from each root u using breadth-first traversal on graph G . With this method, we measure the breadth-first traversal depth D_{uv} for vertex v . A graph's vertices can be grouped into multiple partitions by their depth D_{uv} .

In each partition, we choose the betweenness centrality as labeling criteria to determine the position of each vertex in the final lineup. The betweenness centrality equals to the number of shortest paths from all vertices to all others that pass through the given vertex. We use it in labeling because it shows the importance of each vertex in the graph. For convenience, we use a superscript L to denote a vertex's label, as in v^L . After the labeling procedure, we reorder the vertices in each partition according to their labels and get the final partition.

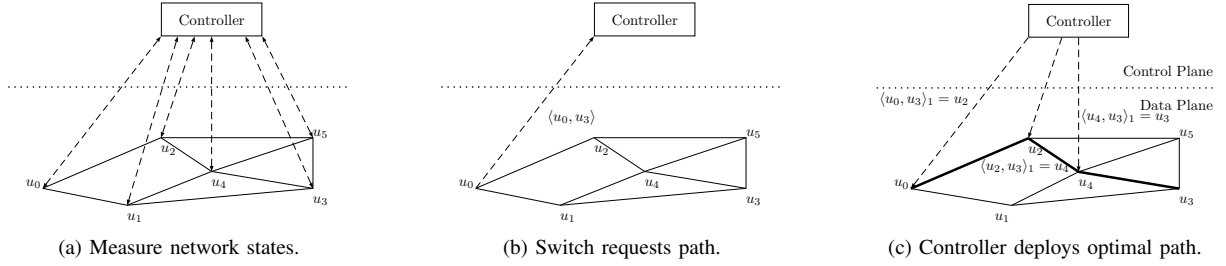


Fig. 1. The proposed routing architecture.

In the convolution part, we first select w vertices as root vertex with the fixed order of vertices indices. For each root vertex, we choose k vertices from its final partition. Subsequently, we reform the input features into matrices of shape $[w, k, k, N_f^e]$ for edge features and $[w, k, N_f^v]$ for vertex features, where N_f^e and N_f^v are the number of input features. Afterward, convolutional kernels of size k^2 and k are applied to these matrices respectively. In practice, we select all vertices as root vertex one by one and explore up to the average path length.

V. PERFORMANCE EVALUATION

A. Experiment Setup

We use Mininet and OpenVSwitch to simulate an SDN environment and the data plane is mapped into six high-performance servers. The OpenVSwitch software switch is customized to encode switch level measurements in packet headers. We also write a controller based on POX to execute proposed routing architecture. In our experiments, we use a real-world topology from AT&T North American backbone network, which has 25 nodes and 112 links. The task is to find an optimal path from each node to another. The controller server is equipped with 4 NVIDIA K80 GPUs.

In experiments, we tested five models. The Deep Belief Architecture (DBA) [11] is presented here as a baseline. The Deep Neural Network (DNN) uses raw network measurements and a non-convolutional neural network. Deep Neural Network with Features (DNNF) is similar to DNN but uses the proposed features. The Convolutional Neural Network (CNN) uses the features and a traditional convolutional neural network. The proposed GADL uses both the proposed features and the graph-aware neural networks.

While an optimal is defined by either latency, throughput, or a combination of both, in experiments we prepare our training data targeting minimal latency paths. The reference algorithm used for preparation is from a recent work [4] which provides provable approximation guarantees.

B. Evaluation Metrics

To start with, since we are using supervised learning to train the models, we adopt the metrics of accuracy from supervised learning to evaluate the learning ability of each model. Let C_i^{actual} be the actual next forwarding node computed from the reference algorithm for sample i , $C_i^{\text{predicted}}$ be the predicted

TABLE I
TIME USED TO TRAIN MODELS AND FINAL ACCURACY

Model	Accuracy	Training Time	Time Used ^a
DBA	0.6921	220.3 minutes	220.3 minutes
DNN ^b	0.0910	—	—
DNNF	0.7207	88.7 minutes	56.7 minutes
CNN	0.8069	65.3 minutes	4.1 minutes
GADL	0.8655	53.6 minutes	1.2 minutes

^aIt is the time for models reaching the same accuracy as DBA.

^bDNN model shows no promising progress after 60 minutes of training. The best result is recorded here.

output from the neural network, and Z be the number of samples tested: $\text{Accuracy} = \text{Count}(C_i^{\text{actual}} = C_i^{\text{predicted}})/Z$. The more close the accuracy is to one, the better the model's learning ability is.

Moreover, we use the cross entropy to monitor the possibility of confusion among multiple available next forwarding nodes when each model is making decisions. The cross entropy is defined as: $\text{CrossEntropy} = -\sum_{x_i} p(x_i) \log q(x_i)$, where for each available choice x_i , $q(x_i)$ is the probability of the model choosing x_i , and $p(x_i)$ is the ground truth whether x_i should be chosen. The more close the cross entropy is to zero, the more confident the model is when making decisions.

At last, we are going to examine the average end-to-end flow latency in testbed with different routing policies. This shows how these learning based models perform in real networking scenarios.

C. Result and Analysis

Table I shows that both DNNF, CNN, and GADL outperform state-of-art DBA in terms of accuracy and training efficiency. All these three models use a centralized architecture in the network controller and they train 3.8X to 183.6X faster than DBA. This is because that a centralized neural network fine-tunes its neuron weights for all pairs of routing tasks rather than training multiple sets of partially overfitted models separately. The boost of performance from DNN to DNNF shows the strength of well-defined features proposed in section IV-A. The features turn raw network measurements into meaningful indicators suitable for routing tasks. Consequently, the deep learning models are able to learn the network dynamics more easily. In general, CNN is superior to DNNF. However,

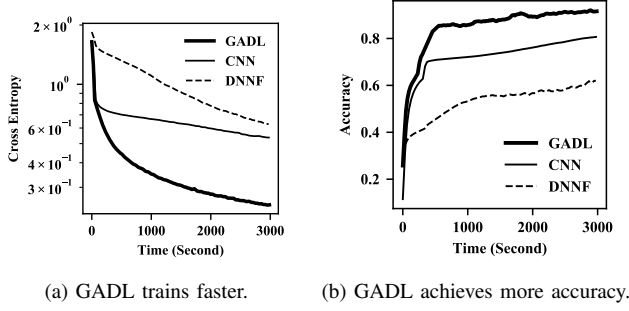


Fig. 2. The training process.

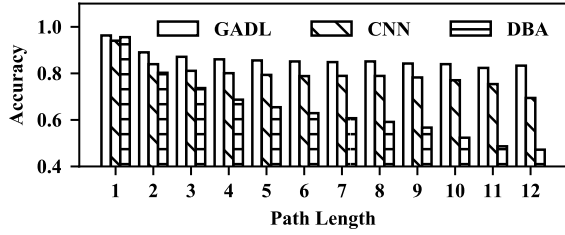


Fig. 3. Model Testing Performance.

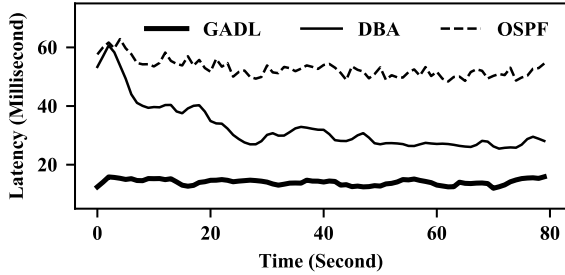


Fig. 4. Average Network Latency.

due to the lack of knowledge of real neighbors in the graph, as mentioned in section IV-B, CNN fails to fully utilize the potential brought by our proposed features. Fig. 2a and Fig. 2b demonstrate that the proposed graph-aware neural network helps GADL to train much faster and better, compared to traditional deep learning techniques in CNN.

Fig. 3 shows our GADL performs well when an optimal path contains multiple hops, which is common in communication networks. In our experiment, optimal paths with length from 2 to 6 covers 83.43 percent cases. The results demonstrate that GADL is smart enough to deduce a multi-hop optimal path in complex routing scenarios. The achievement of GADL benefits from both the designed features and graph-aware structure. In terms of the performance in real networking scenario, Fig. 4 demonstrates that proposed GADL achieves its goal to find optimal paths with minimal latency in complex and variant networking scenes. In our experiments, GADL reduces average network latency by about 50 percent and converges

faster against both OSPF and state-of-art DBA.

VI. CONCLUSION

In this paper, we explained that the advances in software defined networking and deep learning techniques bring the potential of more intelligent routing strategies. A practical solution, however, needs to use a specially customized deep learning architecture for routing. Following the thought, we proposed GADL. The experiments show that the proposed GADL outperforms state-of-art deep learning based routing strategies both in terms of accuracy and efficiency. Our extensive experiments also show that representative features and graph-aware structures are useful in deep learning based network routing. Furthermore, the centralized architecture in software defined networking benefits more from the power of deep learning models.

ACKNOWLEDGMENT

This work was jointly supported by: (1) National Natural Science Foundation of China (No. 61771068, 61671079, 61471063, 61372120, 61421061); (2) Beijing Municipal Natural Science Foundation (No.4182041, 4152039); (3) the National Basic Research Program of China (No. 2013CB329102).

REFERENCES

- [1] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.
- [2] Y. L. Boureau *et al.*, "Sparse feature learning for deep belief networks," in *Advances in Neural Inform. Process. Syst.*, 2008, pp. 1185–1192.
- [3] T. Chen, M. Matinmikko, X. Chen, X. Zhou, and P. Ahokangas, "Software defined mobile networks: concept, survey, and research directions," *IEEE Commun. Mag.*, vol. 53, no. 11, pp. 126–133, 2015.
- [4] M. Chiesa, G. Kindler, and M. Schapira, "Traffic engineering with equal-cost-multipath: An algorithmic perspective," *IEEE/ACM Trans. on Networking*, vol. 25, no. 2, pp. 779–792, April 2017.
- [5] M. Defferrard *et al.*, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Inform. Process. Syst.*, 2016, pp. 3844–3852.
- [6] D. K. Duvenaud *et al.*, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in Neural Inform. Process. Syst.*, 2015, pp. 2224–2232.
- [7] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. of the 14th Int. Conf. on Artificial Intell. and Statist.*, 2011, pp. 315–323.
- [8] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Inform. Process. Syst.*, 2012, pp. 1097–1105.
- [9] Y. Lee, "Classification of node degree based on deep learning and routing method applied for virtual route assignment," *Ad Hoc Networks*, vol. 58, pp. 70–85, 2017.
- [10] Y. Li *et al.*, "Flowradar: A better netflow for data centers," in *13th USENIX Symp. on Networked Syst. Des. and Implementation*, 2016, pp. 311–324.
- [11] B. Mao *et al.*, "Routing or computing? the paradigm shift towards intelligent computer network packet transmission based on deep learning," *IEEE Trans. on Comput.*, 2017.
- [12] N. McKeown *et al.*, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.
- [13] M. Niepert *et al.*, "Learning convolutional neural networks for graphs," in *Int. Conf. on Mach. Learning*, 2016, pp. 2014–2023.
- [14] A. H. Sayed, "Adaptive networks," *Proc. of the IEEE*, vol. 102, no. 4, pp. 460–497, 2014.
- [15] D. K. Sharma, S. K. Dhurandher, I. Woungang, R. K. Srivastava, A. Mohananeey, and J. J. Rodrigues, "A machine learning-based protocol for efficient routing in opportunistic networks," *IEEE Syst. J.*, 2016.