Accepted Manuscript

Learning-based network path planning for traffic engineering

Yuan Zuo, Yulei Wu, Geyong Min, Laizhong Cui

PII: S0167-739X(18)31324-4
DOI: https://doi.org/10.1016/j.future.2018.09.043
Reference: FUTURE 4478

To appear in: *Future Generation Computer Systems*

Received date : 29 May 2018
Revised date : 6 September 2018
Accepted date : 17 September 2018

Available online at www.sciencedirect.com

ScienceDirect

ELSEVIER

Future Generation Computer Systems 00 (2018) 1–10

Journal Logo

www.elsevier.com/locate/procedia

Learning-based Network Path Planning for Traffic Engineering

Yuan Zuo[a], Yulei Wu[a], Geyong Min[a], Laizhong Cui[b],

[a] *College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, EX4 4QF, UK*
[b] *College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, PR. China*

Abstract

Recent advances in traffic engineering offer a series of techniques to address the network problems due to the explosive growth of Internet traffic. In traffic engineering, dynamic path planning is essential for prevalent applications, e.g., load balancing, traffic monitoring and firewall. Application-specific methods can indeed improve the network performance but can hardly be extended to general scenarios. Meanwhile, massive data generated in the current Internet has not been fully exploited, which may convey much valuable knowledge and information to facilitate traffic engineering. In this paper, we propose a learning-based network path planning method under forwarding constraints for finer-grained and effective traffic engineering. We form the path planning problem as the problem of inferring a sequence of nodes in a network path and adapt a sequence-to-sequence model to learn implicit forwarding paths based on empirical network traffic data. To boost the model performance, attention mechanism and beam search are adapted to capture the essential sequential features of the nodes in a path and guarantee the path connectivity. To validate the effectiveness of the derived model, we implement it in Mininet emulator environment and leverage the traffic data generated by both a real-world GEANT network topology and a grid network topology to train and evaluate the model. Experiment results exhibit a high testing accuracy and imply the superiority of our proposal.

Keywords: Traffic Engineering, Path Planning, Deep Learning, Sequence-to-sequence

1. Introduction

Traffic Engineering (TE) is crucial for enhancing the utility and performance of networks in the era of big data [1, 2]. It has been a hot topic in recent research trend [1, 2, 3, 4, 5], due to the increasing demands on dynamic network maintenance, management and Quality-of-Service (QoS) guarantee [2].

Path planning is one of the most important aspects of TE, providing selected routing and forwarding paths between nodes to offer high-performance networks [6, 7]. Due to diversified services and advanced networking techniques like network virtualization, network dynamics have become a normal phenomenon, resulting in an increasing demand for path planning under various requirements and/or conditions [7, 8, 9, 10]. For instance, forwarding path is restricted to go through one or several given nodes for e.g. traffic monitoring.

The emerging Software Defined Networking (SDN) paradigm [11, 12] has gained its popularity and drawn considerable attentions for smooth and rapid development of new TE algorithms, due to its capability of decoupling the control plane from the data plane. This decoupling releases computing resources in commodity switches to simplify switch functions and pull the forwarding decision making and computing into a high level controller, which is able to take global information into consideration.

With the increment of various emerging applications, massive data has been produced from the Internet [13, 14]. The hidden information behind the data implies important knowledge [15] for efficient TE [3]. Data-driven applications take as input the massive data to help adapt learning algorithms into distinctive circumstances. Successes in [16, 17] clarify the importance of leveraging useful data generated from the network. Their large-scale data analysis enables promising applications, such as semantic interpretation and user experience improvement.

We intend to extend data-driven learning promoted by deep learning methods to the path planning problem. Deep learning has been massively developed and rapidly deployed by a variety of applications [18], which makes the utmost data be fully perceived by the application. Though deep learning methods have succeeded in many fields, challenges still exist when being introduced into network applications [3]. For example, dynamics in networking needs human-like behavior for model to fit. To solve this problem, the authors in [4, 5] incorporated reinforcement learning to flexibly make decisions in a dynamic network environment, but they focus on coarse flow split ratio control for communication transferring.

To make a further progress, in this paper we aim to have finer-grained network-level path planning driven by empirical traffic data. To achieve this purpose, we treat the network path as a sequential data, since a path with a set of forwarding hops explicitly express its serializability. One notable sequence analysis technique comes from Natural Language Processing (NLP), where sentences or phrases are intrinsically serialized. Inspired by sentences analysis in NLP, we merge neural networks and build a sequence-to-sequence (seq2seq) model [19] to capture inner characteristics of sequence-like traffic forwarding and routing path. Abundant traffic information from the network can provide a natural way for model training. We will extract sequential features from empirical traffic data and apply them into path discovery.

The main contributions of this paper can be summarised as follows:

- This paper proposes a learning-based data-driven model for network planning under constrained conditions, which can be used in SDN controllers to perform finer-grained TE. The model is derived based on the seq2seq processing model with empirical knowledge from historical traffic data. With the derived model, we can have a sequence of nodes as input and predict a sequence of nodes as target path.

- In order to infer a reasonable order of nodes in the target path, attention mechanism is adapted to calculate the correlations between input sequence and target sequence.

- To further ensure the connectivity of the output target path from the derived model, we apply beam search to extensively search candidate sequences. Beam search is able to help the prediction of target paths move out of local optima and explore global optimal solutions. The model can check the order of nodes in terms of the connectivity of a path and only output the available ones.

- To validate the effectiveness of the derived model, we implement it in a typical SDN emulator environment, i.e., Mininet, and leverage the traffic data generated by both a real-world GEANT network topology and a grid network topology to train and evaluate the model. Experiment results exhibit a high testing accuracy and imply the superiority of our proposal.

The remainder of the paper is organized as follows. In Section 2, related work is described. Section 3 introduces preliminary work. A detailed description of our proposed model is presented in Section 4. Experiment results and analysis are conducted in Section 5. Finally, we conclude this study in Section 6.

2. Related Work

Path planning has been widely studied in wireless networks [8, 9, 10] for producing optimal forwarding paths under constraints. Especially, it is crucial for energy-sensitive applications in e.g. wireless sensor networks to meet resource constraints. Most of existing algorithms have been designed at application-level.

Although application-level designs can surely improve network performance, they are not optimised based on general networking scenarios but are based on limited application situations, e.g., leveraging two classic layered approaches in [8]. In contrast, the network-level approaches are finer-grained and can be generalized to different network architectures.

In [20], the authors studied and proposed dynamic routing algorithms for online unicast and multicast requests. The authors considered switch bandwidth, link capacity and bandwidth demand together to maximize accumulated bandwidth of admitted requests. They separated the problem into two issues, link resource cost regression and link capacity maximization without the knowledge of previous request arrivals.

Encoding path is essential for forwarding table state maintained in a switch [21]. To relieve the pressure on the maintenance of switch tables, the authors in [21] proposed a path encoding algorithm to handle variable length interface labels that can shorten any network paths.

A dynamic routing framework based on SDN controller was described in [22], which is to provide the solution for achieving the maximum throughput and minimum cost. The authors used an LSTM model to predict traffic matrix, based on which the routing rules are generated by traffic routing units.

Very recently, in [5], the authors aimed to schedule conventional dynamic traffic and Internet of Things (IoT) with flexible time. They proposed to integrate the reinforcement learning into the decision of IoT traffic transmission. They firstly identified the inefficiency for IoT transmission, and then built a learning model to formulate IoT traffic scheduling issue and determine a controlled split ratio for it. The reward function was given by weighted sum of three types of traffic volumes.

In [4], Chinchali et al. gave a clear definition of the targeted Traffic Engineering (TE) problem, and successfully built a model-free reinforcement learning for a general dynamic routing framework. Two key methods were included, an actor-critic model with experience replay training and a TE-aware exploration, which absorbs the trade-off policy of exploration and exploitation in reinforcement learning. In addition, they utilized the network utility, formulated in [23], of throughput and delay as the reward. Their empirical experiments showed promising results for applying of reinforcement learning in TE problems.

In spite of the successful integration of deep learning, as aforementioned, they only focused on high level flow-oriented split ratio optimization [5], and assumed K available candidate solutions prior to dynamic decisions [4]. In contrast, in this paper we will consider a finer-grained traffic learning and achieve network-level path planning.

3. Preliminaries

The seq2seq model [19] is an expanded and specific encoder-decoder model in neural networks for handling sequence data, including sequence data learning, transferring and translation. It now prevails in neural machine translation, text summarization and speech recognition in Natural Language Processing (NLP), image captioning and other sequence data applications. In this section, we give an overview of the formulation and the details of the seq2seq model. To facilitate the understanding, the basis of neural networks and the encoder-decoder structure are introduced in Sections 3.1 and 3.2, respectively, followed by the seq2seq model in Section 3.3.

3.1. Basis of neural networks

This study concentrates on building a prototype neural network model to extract and restore hidden optimal paths between network nodes based on empirical forwarding trace data. In this section, we introduce fundamental concepts of prevalent neural network models as background knowledge.

3.1.1. Artificial neural network

As is well known that neural network is deeply inspired by the biological structure of human brain, an artificial neural network consists of a collection of neurons (nodes), connections (weights), and activations (linear or non-linear functions). Fig. 1 illustrates the standard architecture of an artificial neural network.

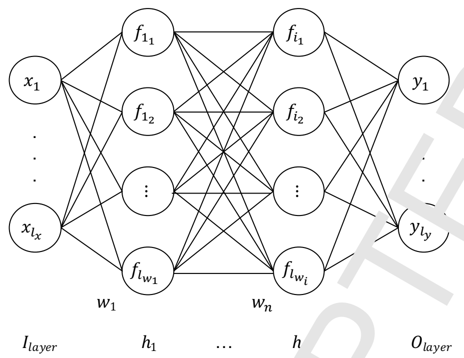


Fig. 1. The standard architecture of a fully connected neural network.

In Fig. 1, an n-layer neural network model is presented. Technically, three types of layers are involved: input layer, I_{layer}, hidden layer, h_i, where $i = 1, 2, 3, ..., n$ and output layer, O_{layer}. Essentially, the size of input layer equals to the dimension of input data, and the size of output layer matches the dimension of output layer. As for the size of hidden layer, it depends on the choice of specific applications. Let $\overrightarrow{x}$ with dimension l_x represent the inputs, $\overrightarrow{w_i}$ with dimension l_{w_i} denote the weights in the i-th hidden layer, $f_i(\overrightarrow{w_i}, f_{i-1})$ with the inputs of the previous one f_{i-1} be the outputs of the i-th hidden layer, and $\overrightarrow{y}$ with dimension l_y represent the outputs of the output layer.

Depending on specific circumstances, the choice of the function f of hidden layer outputs could be linear or non-linear activation functions:

- $\sigma(\cdot)$: sigmoid or logistic function;
- $tanh(\cdot)$: hyperbolic tangent function;
- $Relu(\cdot)$: rectified linear unit [24].

The softmax function is widely used as the activation function of the output layer for multi classification, and sigmoid is usually used for binary classification.

3.1.2. Recurrent neural network

A noticeable drawback of basic artificial neural network is that it cannot model sequential data which are interrelated orderly. In this section, we introduce a variant structure, Recurrent Neural Network (RNN), which is particularly devised for sequential modeling in neural networks. RNN has gained its popularity and shown promising performance thanks to its capability of taking information from past to subsequent inputs.

In detail, RNN attempts to *memorize* sequential histories and merge them into current observation in order to predict the next sample element in the same sequence. The "*memory*" unit is usually called *cell*. A simple architecture of RNN is presented in Fig. 2.

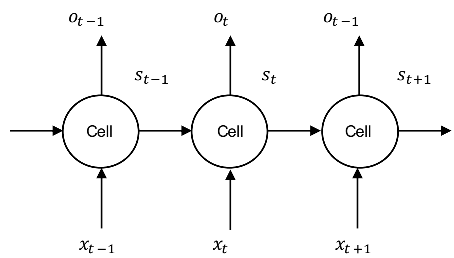


Fig. 2. A simple architecture of RNN.

In Fig. 2, it can be seen that at time step t, an input sample x_t is pushed into a cell (described below), based on which the hidden state s_t is generated. Ultimately, the output o_t would be given by the hidden state s_t.

The hidden states and outputs can be represented in a mathematical way as follows:

- $s_t = f(Ux_t + Ws_{t-1})$, where U is an input matrix and W is the weights similar to the basic artificial neural network;
- $o_t = softmax(Vs_t)$, where V is an output matrix mapping the hidden state s_t into classification scores;
- $f(\cdot)$ is considered as the core component, widely accepted as the basic cell $tanh(\cdot)$, Long Short Term Memory (LSTM) cell [25] and Gated Recurrent Unit (GRU) cell [26].

It is worth noting that the LSTM structure overcomes long sequence learning and gradient vanishing problems with input, output, and forget gate to sift and keep key information. It is suitable for memorizing long-term forwarding hops and forgetting unnecessary interference caused by predecessors. Therefore, we choose LSTM cell in our work.

3.2. The Encoder-Decoder structure

The goal of Encoder-Decoder structure is regarded as a mechanism to map the data in the source space into the desired information in the target space via an intermedia space. Specifically, two parts are included: *Encoder* and *Decoder*. Let x and y denote the data in the source space and target space, respectively, and m represent the data in the intermedia space. Therefore, we have

$$Encoder : \overrightarrow{m} = En(\overrightarrow{x}), \forall \overrightarrow{x} \in D_s, \forall \overrightarrow{m} \in D_m$$

$$Decoder : \overrightarrow{y} = De(\overrightarrow{m}), \forall \overrightarrow{y} \in D_t, \forall \overrightarrow{m} \in D_m$$

where D_s is the source space with dimension l_x, D_m is the intermediate space with dimension l_m, and D_t is the target space with dimension l_y.

The Encoder-Decoder structure is able to learn the source space knowledge to align with the desired target space knowledge, referred as "*translation*" or "*transduction*" in some cases.

Note that the setting of $D_s = D_t$ ensures that our constrained path planning problem in traffic engineering, consiering the request pairs (source -> destination nodes) and the constrained conditions (source -> constraints -> destination), can be fitted into this scheme, because they are in the same objective space as controlled network nodes.

3.3. The sequence-to-sequence model

The seq2seq model was proposed in [19] for neural machine translation, which now is extended as general-purpose sequential model in many other spheres, for instance, conversational modeling, image captioning, etc. An evident advantage of seq2seq model is that it can encode variable length sequences into a fixed-length coding vector bridging the gap between source space and target space.

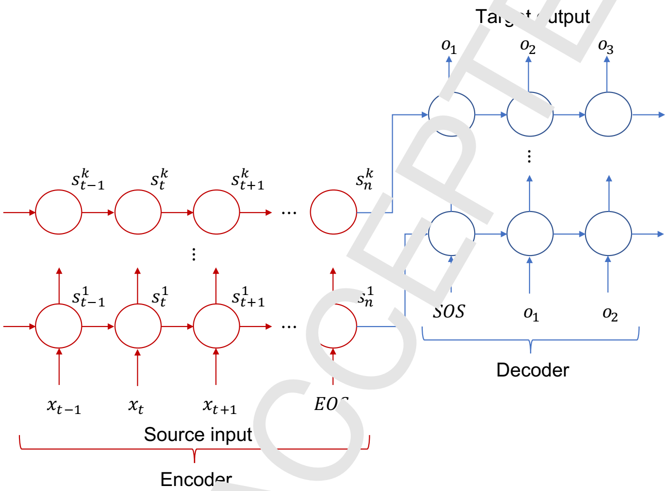


Fig. 3. An abstract structure of seq2seq model.

A illustration of the seq2seq model is depicted in Fig. 3. As can be seen, two parts are included: *Encoder* marked red and *Decoder* marked blue. With the source input $\overrightarrow{x}$, adding an ending symbol, EOS, there can be stacked by multiple layers of the RNN instances in *Encoder* compressing ordered information into final hidden states, which is then sent to the counterparts in *Decoder*. On the opposite side, *Decoder* takes the hidden states and the estimated outputs, adding a starting symbol, SOS, as the initialization and inputs of its corresponding RNN instances, respectively.

4. The Proposed Framework

This study aims to propose a forwarding method based on seq2seq model for path planning between two nodes, through learning common paths in a network from historical forwarding experiences.

4.1. Problem formulation

For the sake of clarity of illustration, Definition 1 is first given in this section.

Definition 1. *The source sequence and target sequence are two network paths between source node and destination node. Each of them contains a set of network nodes to be traversed from source to destination.*

The problem we considered in this paper is that, given the source sequence and constrained condition, we are going to find the target sequence which satisfies the constrained condition. For example, a source sequence can be (source, node 1, node 2, ..., node n, destination), and a constrained condition is that this network path need go through a particular node k. The target sequence is therefore (source, node 1, node 2, ..., node k, ..., node n, destination).

For clarity, we denote source and target sequences as bold vector $\overrightarrow{\mathbf{x}}, \overrightarrow{\mathbf{y}}$, where $\overrightarrow{\mathbf{x}} = (x_1, x_2, ..., x_{n_x})$ with length n_x and $\overrightarrow{\mathbf{y}} = (y_1, y_2, ..., y_{n_y})$ with length n_y. The lower case letter x, y represent the element in a sequence.

4.2. The proposed forwarding method

Fig. 4 provides a high-level illustration of the proposed forwarding method based on seq2seq model. The *Encoder* takes the source sequence as input and produces hidden states that are fed into the *Decoder* and the *Attention* (used to ensure reasonable order of nodes in a sequence). The *Decoder* receives the last hidden state from the *Encoder* to produce its own hidden states which are fed into the *Beam search* (used to enhance the performance of the proposed model) and back to the *Attention*, as the dashed arrow shows. The *Attention* scores the relevance between the hidden states of the *Encoder* and the *Decoder* to output context vectors. Eventually, the context vectors are jointly fed into the beam search with the hidden states of the *Decoder*, which is represented by the *addition* icon in the figure. The details of *Encoder* and *Decoder* have been illustrated in Section 3.2, and the details of *Attention* and *Beam search* will be elaborated in Sections 4.3 and 4.4.

Let the Greek letters θ, ω represent parameters and weights in neural networks, and let the capital letter $\mathcal{D}_s, \mathcal{D}_t$ denote the dataset of source sequence and target sequence, respectively. The number of elements in $\mathcal{D}_s$ and $\mathcal{D}_t$ is k.

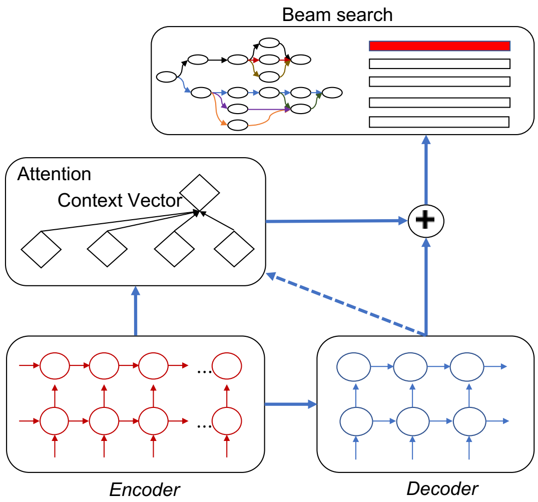


Fig. 4. A high-level illustration of the proposed forwarding method based on seq2seq model

As described in Section 3.2, we have the form of *Encoder* below:

$$\overrightarrow{\mathbf{m}_i} = En(\overrightarrow{\mathbf{x}_i}, \theta), \quad \overrightarrow{\mathbf{x}_i} \in \mathcal{D}_s, \quad i = 1, 2, 3, ..., k \tag{1}$$

where $\overrightarrow{\mathbf{x}_i}$ is one of k elements in the dataset $\mathcal{D}_s$, and θ is the parameters of *Encoder*.

The source sequence is encoded into a fixed-length (dimension) vector as an intermedia sequence to bridge the gap (in terms of the number of network nodes) between a source sequence and a target sequence.

Equivalently, the *Decoder* can be given by:

$$\begin{gathered} y_i^j = De(\overrightarrow{\mathbf{y}_i^{-j}}, \omega|\overrightarrow{\mathbf{m}_i}), \quad \overrightarrow{\mathbf{y}_i} \in \mathcal{D}_t \\ i = 1, 2, 3, ..., k, \quad j = 1, 2, 3, ..., n_y \end{gathered} \tag{2}$$

where $-j$ indicates a sub-sequence $(y_1, y_2, ..., y_{j-1})$ of $\overrightarrow{\mathbf{y}_i}$ before the j-th element appears, and ω is the parameters of *Decoder*.

For simplicity, Eqs. (1) and (2) can be re-written as follows:

$$\begin{gathered} y_i^j = \mathcal{F}(\overrightarrow{\mathbf{y}_i^{-j}}, \theta, \omega|\overrightarrow{\mathbf{x}_i}), \quad \overrightarrow{\mathbf{x}_i} \in \mathcal{D}_s, \quad \overrightarrow{\mathbf{y}_i} \in \mathcal{D}_t, \\ i = 1, 2, 3, ..., k, \quad j = 1, 2, 3, ..., n_y \end{gathered} \tag{3}$$

Normally, $\overrightarrow{\mathbf{x}_i}$ is a two-element sequence, source x_{src} and destination x_{dst} for basic forwarding. Meanwhile, with a constrained condition, $\overrightarrow{\mathbf{x}_i}$ will include particular network nodes, e.g. node IDs, through which a flow should go. This can be expressed as:

$$\begin{gathered} y_i^j = \mathcal{F}(\overrightarrow{\mathbf{y}_i^{-j}}, \theta, \omega|x_{src}, x_{res}, x_{dst}), \\ x_{src}, x_{res}, x_{dst} \in \mathcal{D}_s, \quad \overrightarrow{\mathbf{y}_i} \in \mathcal{D}_t, \\ i = 1, 2, 3, ..., k, \quad j = 1, 2, 3, ..., n_y \end{gathered} \tag{4}$$

It is worth noting that x_{res} represents a single restricted node, whereas a set of restricted nodes can be added, namely, $\overrightarrow{x_{res}} = \{x_{res1}, x_{res2}, ...\} \subset \mathcal{D}_s$.

Because the sequence output from standard seq2seq model could be variant in statistics due to flexibility and randomness, the network path represented by the target sequence, y_i^j, may encounter the problem of non-connectivity. This is not tolerant in the forwarding method for path planning in traffic engineering. To effectively alleviate this issue, the attention mechanism [27, 28] and beam search [29] will be employed. The attention mechanism is an enhancement for context relevancy learning, and the beam search tops the best score and attempts to ensure the link connectivity. The details of leveraging attention mechanism and beam search can be found in Sections 4.3 and 4.4. The situation of rarely happened non-connectivity path in the target sequence after the adoption of attention mechanism and beam search is discussed at the end of Section 4.4.

4.3. Attention mechanism

The attention mechanism, also named alignment, has been applied in [27, 28], aiming at aligning the elements of a sentence or a phrase in a correct order in neural machine translation. In this study, it is adapted to ensure reasonable sequence order by scoring the relevance of the elements between source space and target space.

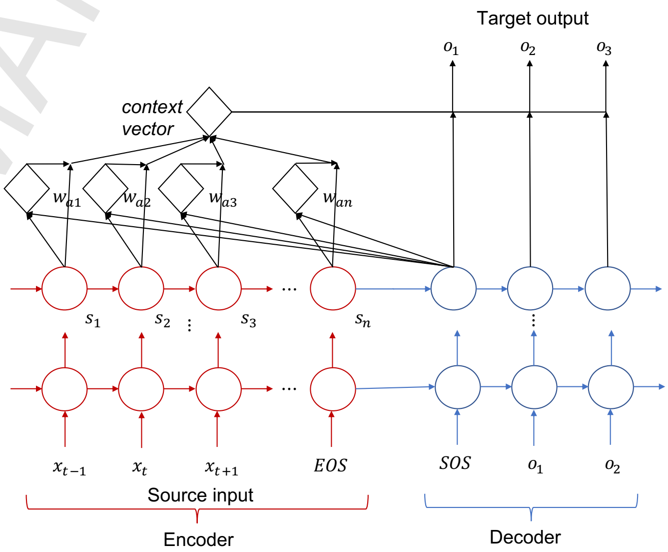


Fig. 5. An illustration of attention mechanism

Fig. 5 illustrates the basic idea of attention mechanism. Instead of directly using encoder hidden states, the attention mechanism sums the encoder hidden states as the context vector by weights w_{ai}, which are scored with decoder hidden states. Thus, along with the context vector, the model also takes hidden states of the last decoder layer as comprehensive information to obtain the final target outputs. The score function could be multiplicative or additive, depicted in [27, 28].

Because the attention mechanism captures key relevance, we take advantage of it to effectively restore the complete restricted forwarding paths. The *Decoder* can therefore use the context vector $\overrightarrow{\mathbf{c}}$ with the last decoding hidden state $\overrightarrow{\mathbf{h}_D}$ as follows:

$$\varphi_i = \mathcal{F}_{Attention}(h_{E,i}, h_{D,t-1}) \quad \forall i, 1 \le i \le n_x \quad \forall t, 1 \le t \le n_y \tag{5}$$

$$\alpha_i = softmax(\varphi_i) \tag{6}$$

$$\overrightarrow{\mathbf{c}_t} = \sum_i \alpha_i \cdot h_{E,i} \tag{7}$$

where the relevance score between *Encoder* hidden states $h_{E,i}$, $1 \le i \le n_x$ and the *Decoder* previous step hidden state $h_{D,t-1}$ is denoted as φ_i. α_i is the weight for *Encoder* hidden states to compute weighted sum of the context vector, $\overrightarrow{\mathbf{c}_t}$.

The final output of target sequence with the context vectors taken into account can be expressed as:

$$y_i^j = \mathcal{F}(\overrightarrow{\mathbf{y}_i^{-j}}, \theta, \omega, \overrightarrow{c_i} | \overrightarrow{\mathbf{x}_i}) \tag{8}$$

$$y_i^j = \mathcal{F}(\overrightarrow{\mathbf{y}_i^{-j}}, \theta, \omega, \overrightarrow{c_i} | x_{src}, x_{res}, x_{dst}) \tag{9}$$

where Eq. (8) shows the output without constrained condition, and Eq. (9) denotes that with constrained condition taken into account.

4.4. Beam search

The aforementioned LSTM cell, the seq2seq structure and the attention mechanism enormously facilitate sequential data sensing and cognition. However, the single output result (i.e. only one target output) may cause local optimum. Especially, sequential models normally cannot rerun, failing to circumvent temporary single highest score and gain rewards in the long run. A potential solution is to collect as many candidate traces as possible to avoid this local optimum problem. Hence, in this study the beam search algorithm [29] is adapted to achieve this purpose and boost the performance of the proposed model.

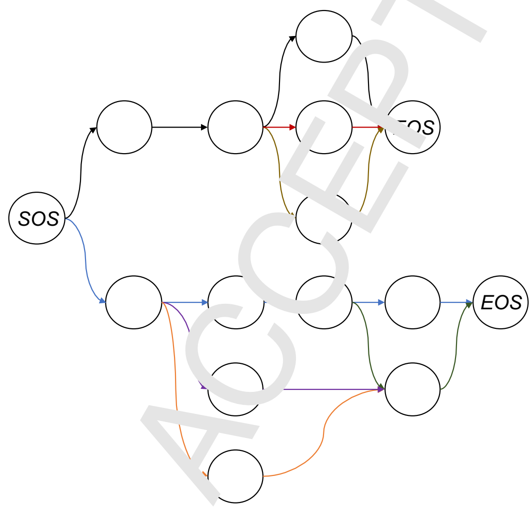


Fig. 6. An example of beam search with beam width-7

The idea of beam search is to widen the model search range by buffering n traces, which provides a list of outputs by scoring the sequence context. n is the beam search width. A simple example is presented in Fig. 6, where 7 paths are drawn with 7 colors, marked with starting symbol, SOS, and ending symbol, EOS. Beam search can efficiently explore the target space and output the top-n paths against the single result. Compared to breadth search, which exhausts all options, beam search eliminates highly unlikely traces to accelerates training and testing.

In this study, beam search is adapted in the *Decoder* not only for providing optimal target output, but also for presenting the target output as valid forwarding path. Let beam search width n be 5, a list of top 5 sequences are therefore buffered. From the top 1 to the last (i.e., 5 in this case), any duplicate nodes are eliminated canceling wasteful loops, and checking whether the successor node is a neighbor of the previous node to verify the connectivity of forwarding path of the output target.

It is worth noting that the consideration of attention mechanism and beam search can still miss a very small fraction of path validity (path non-connectivity) from the empirical results. As a matter of the fact, we can conduct the last guarantee via recomputing the non-connectivity path. For example, if all 5 candidates lose their validity, we choose the top 1 and recompute the path between the disconnected node and the destination. In practice, this situation may lead to long delay, however it occurs very rarely.

5. Experiments and Analysis

In this section, we conduct experiments to evaluate the effectiveness of the proposed learning-based path planning model under constrained conditions. SDN is used as the network environment due to its popularity and pervasive application in traffic engineering [2]. In this section, we employ the Mininet emulator [30], and deploy our proposed learning-based forwarding model in POX, a typical and popular implementation of SDN controller, to make decisions for path planning under constrained conditions. Mininet emulates a virtual network with a set of virtual hosts that can run various network services. It is specifically designed for SDN scenario embedded with OpenFlow specification.

In what follows, we will show the environment settings, results and discussions of our experiments. The experiments consist of two major parts:

1. The training and evaluation of seq2seq model
2. SDN network simulation employing the model

5.1. The sequence-to-sequence model

The seq2seq model is built on top of the TensorFlow library published by Google [31]. Two types of network topology are selected in our evaluation: the 2012 Europe GEANT network topology with 40 nodes, shown in Fig. 7, and a 10x10 Grid network topology with 100 nodes, shown in Fig. 8. We also constrain the bandwidth between switches [1] as 50Mbps and link

[1]In SDN, network nodes are called switches.

delay as 2ms. Meanwhile, to eliminate the impact of connection between host and switch on network performance, the host-switch link bandwidth is set to be 1000Mbps with 2ms link delay. In our experiments, we set the constrained condition, $\overrightarrow{x_{res}}$, to be one-node and multi-node (i.e., two nodes), respectively. Namely, the planned network path between source and destination need go through one particular node (one node constrained) and two particular nodes (multi-node constrained) in the network. In the following, we will show training data preparation and the training and evaluating of the proposed model, for each of the two topologies.

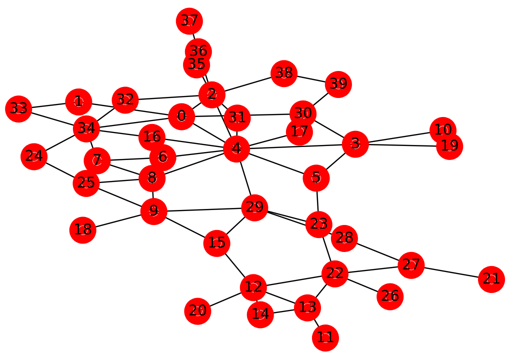


Fig. 7. The topology of Europe GEANT network

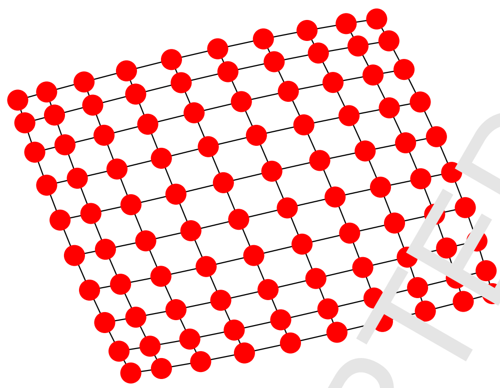


Fig. 8. The topology of 10×10 grid network

- **Training data preparation**

To collect the training data, we first implement the Dijkstra's shortest path algorithm to generate a collection of static paths between nodes in a network. The shortest paths between nodes are usually not unique, but we focus on the unicast scheme and extract only one for each source and destination pair. These paths are considered as *optimal experiences*. In addition to these optimal experiences, we shall generate the *restricted experiences* as the experiences for traffic forwarding under constrained conditions.

First of all, the data preparation for one-node constraint (i.e., the path between source and destination need go through a given node) is detailed below, followed by the multi-node constraint (for the convenience of presentation, two-node constraint is considered).

To obtain the restricted experiences, all network nodes will get chance to be selected as the restricted node (one is selected each time) except for the ones that have already been in the paths collected for optimal experiences and those that lead to routing loops (the examples will be provided below to illustrate this routing loop situation). In order to form the *restricted paths* (the paths collected for restricted experiences) with the one-node constraint, two separated paths are calculated: the one between the source and the constrained node and the one between the constrained node and the destination, respectively. Then, these two paths are concatenated as the restricted path.

Let us use the example below in the GEANT network topology to illustrate how the restricted path is collected. We consider two scenarios of traffic forwarding, each with one constrained node. The first scenario has node 1 as the source, node 5 as the destination, and node 2 as the constrained node, i.e., $x_{src} = 1$, $x_{res} = 2$, and $x_{dst} = 5$, represented by $(1, 2, 5)$. A second scenario has $x_{src} = 1$, $x_{res} = 16$, and $x_{dst} = 5$, denoted by $(1, 16, 5)$. The two separated paths for $(1, 2, 5)$ are:

$$(1, 2) \Rightarrow (1, 0, 2), \quad (2, 5) \Rightarrow (2, 0, 4, 5),$$

and the two separated paths for $(1, 16, 5)$ are:

$$(1, 16) \Rightarrow (1, 33, 34, 16), \quad (16, 5) \Rightarrow (16, 4, 5).$$

The path for the scenario of $(1, 2, 5)$ will not be collected as the restricted path for restricted experiences, because a routing loop is involved in this path due to the overlapped node 0 in these two separate paths. In contrast, the path for the scenario of $(1, 16, 5)$ will be collected as the restricted path for restricted experiences.

One sample data in the dataset for model traning is a pair of source and destination and the path for the target sequence:

$$\{(1, 5), (1, 0, 4, 5)\},$$

or

$$\{(1, 16), (1, 33, 34, 16)\}.$$

When generating the restricted paths for restricted experiences, we also consider the number of hops in the network, i.e., we only collect the paths whose length in terms of hops between source and destination is equal to or less than 20. Table 1 shows the size of collected data based on the length of paths. Note that there is no big difference between the data size when considering different path lengths in GEANT network topology. That is because most of the forwarding paths between any random source and destination pair are close to 10 hops. Therefore, in the experiments, we train and test the proposed model using GEANT network topology with the path length of 20, as it contains the cases of path length of 10 and 15 hops.

Table 2 presents the data size for training and testing dataset split by the ratio of 80%:20%. Note, as mentioned above, only the path length of 20 in GEANT network is used for training.

On top of one-node constraint dataset, two-node constraint dataset is easily constructed and the path representation is very

Table 1. Data size by path length of one-node constrain

	Length = 10	Length = 15	Length = 20
GEANT	16094	17946	17962
Grid	89086	290424	421844

Table 2. Data size of the training and testing set of one-node constrain

		Length = 10	Length = 15	Length = 20
GEANT	Training			14369
	Testing			3593
Grid	Training	71269	232339	337475
	testing	17817	58085	84369

similar. We firstly select a source and destination pair as our objective, then search the extracted one-node constraint dataset to pick all paths that share the same source but are not embedded into its shortest path. Finally, we concatenate the one-node constrained path with the destination as the two-node constrained path. An example is presented below:

Suppose we want the pair $(1, 3)$ to be a two-node constrained path. In one-node constrained path dataset, we have the path of $(1, 16, 5)$ as $(1, 33, 34, 16, 4, 5)$. It is easy to check that $(1, 16, 5)$ is not a segment of the shortest path of $(1, 3)$. As such, $(1, 16, 5, 3)$ shall be the two-node constrained path of the pair $(1, 3)$,

$$(1, 16, 5) \Rightarrow (1, 33, 34, 16, 4, 5), \quad (5, 3) \Rightarrow (5, 3)$$

and the complete path shall be:

$$(1, 16, 5, 3) \Rightarrow (1, 33, 34, 16, 4, 5, 3)$$

The dataset size is shown in Table 5.

- **Training and evaluating the proposed model**

As has been described in the above sections, we set the hyper-parameters of the proposed model as follows: in the *Encoder*, two stacked bi-directional LSTM layers are adopted, which can be regarded as four layers. As opposite, the *Decoder* has four uni-directional LSTM layers. The size of hidden layer is 1024, the embedding size is 100 for nodes (e.g., their IDs) mapping into a vector space, and the beam search width is 5. The model is trained using the mini-batch method with batch size 100 and the Adam optimizer [32]. The experiments are run in a server with Intel 24-core Xeon E5-2650 CPU, 32GB memory and GTX GeForce 1080Ti. The training results of GEANT network topology and grid network topology are shown in Table 3 and Table 4, respectively.

Table 3 indicates that beam search helps gain better performance of the model. The beam width of 1 refers to the situation that only the first outcome is taken into consideration, while beam width of 5 refers to the situation that the model will output 5 candidate paths, and if there is any disconnected path found, the model will move to the next candidate and perform the same check.

Table 3. The accuracy of the training and evaluating of the proposed model in the GEANT network

		Beam=1 Length=20	Beam=5 Length=20
GEANT	Training	0.9794	1.0
	Testing	0.9740	0.9991

Table 4. The accuracy of the training and evaluating of the proposed model in the grid network

		Beam=5		
		Length=10	Length=15	Length=20
Grid	Training	1.0	0.9990	0.9998
	Testing	0.9901	0.9984	0.9987

An intriguing discovery of path inference is worth to note. Since the dataset contains non-restricted paths (shortest paths in this study) and restricted paths, and it is evenly randomly split into training and testing set, the model might not witness the shortest path, i.e., $\{(1, 5), (1, 0, 4, 5)\}$ may only occur in testing data. The model is somehow still able to discover the corresponding non-restricted path. That is $\{(1, 5), (1, 0, 4, 5)\}$ is not witnessed in training, nevertheless the model can still output $(1, 0, 4, 5)$ with the input $(1, 5)$. We conjecture that the neural network is capable of extracting sub-sequential structure embedded in a super-sequence. This feature implies that the proposed model can effectively capture correct paths with partial experiences. Another fact is there is still a tiny fraction of paths that cannot be covered to guarantee their connectivity. We conduct a final check to fix this problem as discussed in the end of Section 4.4.

Furthermore, we extend our experiments to two-node constraint as an example of multi-node constraint, where the collection is based on one-node constraint, as described above.

Table 5. The accuracy of the training and testing in two-node constraint experiments and dataset size, beam width=5.

	Training	Testing	Data Size
GEANT	1.0	0.9999	154663
Grid	0.9993	0.9999	4038738

Table 5 shows the accuracy of the training and testing with two-node constraint experiments. The data sizes of two network collections, GEANT and Grid topology, are 154663 and 4038738, respectively.

5.2. Experiment results and analysis

In what follows, we will introduce the details of the experiment results conducted by our proposed model. In order to show the performance of our proposed model, we use the result from the model for the network without any constrained conditions as the baseline. The results of one-node constraint are shown in Table 6 and Table 7, respectively, for GEANT topology and grid topology.

We set up a host for each switch to generate packets. 100 source-destination pairs are randomly selected. Clearly, the 100 pairs interfere with each other heavily, which causes frequent request conflicts and bandwidth competition. Hence, the network congestion happens frequently as well.

Table 6. Experiment results of one-node constraint in the GEANT network

		Throughput	Congest Delay	Delay
Non-res		1136.15Mbps	295.45ms	15.62ms
Res	100%	711.48Mbps	343.36ms	23.21ms
	50%	908.04Mbps	312.49ms	20.02ms
	20%	1048.17Mbps	307.03ms	16.74ms

Table 7. Experiment results of one-node constraint in the grid network

		Throughput	Congest Delay	Delay
Non-res		1267.67Mbps	329.13ms	25.71ms
Res	100%	781.39Mbps	344.29ms	41.15ms
	50%	1105.09Mbps	380.40ms	33.00ms
	20%	1217.33Mbps	364.47ms	36.76ms

In Table 6 and Table 7, three metrics of network performance are presented, i.e., average throughput, delay in congested condition, and delay in non-congested condition. "Non-res" represents the network without any constrained conditions, and "Res" denotes the network with constrained conditions. The percentage in "Res" indicates the volume of traffic that are forwarded by restricted paths. Note that, as our main work is focusing on proposing a learning-based forwarding strategy, at this point, the forwarding strategy is to randomly choose a constrained node without taking the performance metrics into account. It is reasonable that the random pick might deteriorate the throughput because the paths may lead to unexpectedly long path. It therefore overlaps with more other traffic paths, which triggers more traffic congestion.

As can be seen from the results shown in Table 6 and Table 7, despite the performance being polluted under 100% restricted paths, it is not affected heavily for regulating a part of traffic in both topologies. 20% restrictions show very promising throughput since it is very close to baseline result (network performance under no constrained conditions). As for delay of the network with constrained conditions, both congested and non-congested conditions have no big difference compared with the network without any constrained conditions, emphasising the superiority of the proposed model.

For two-node constraint experiment results, in Table 8 and Table 9, three experiments are conducted, measured by three performance metrics.

Table 8. Experiment results of two-node constraint in the GEANT network

		Throughput	Congest Delay	Delay
Non-res		1136.15Mbps	295.45ms	15.62ms
Res	100%	494.32Mbps	37[illegible].97ms	29.08ms
	50%	877.31Mbps	358.21ms	22.27ms
	20%	1076.22Mbps	338.40ms	18.24ms

Table 9. Experiment results of two-node constraint in the grid network

		Throughput	Congest Delay	Delay
Non-res		1267.67Mbps	329.13ms	25.71ms
Res	100%	690.6[illegible]Mbps	378.26ms	40.01ms
	50%	1[illegible]0.25Mbps	358.82ms	31.67ms
	20%	1[illegible]42.23Mbps	390.22ms	28.79ms

For GEANT topology, the throughput of 100% one-node constraint paths is significantly higher than that of two-node constraint paths as well as the counterpart of grid topology. However, in 50% situation, the throughput of two-node constraint are just slightly below the one-node case, in both two network topologies. It is worth noting that the throughput of two-node constraint is slightly higher than that of one-node situation. This may indicate that the network transferring with more hops do not always cause more interference. We conjecture that some of two-node constraint paths truly circumvent local busy cliques, which trades off delay but reduces routing conflicts. It also brings the chance that reasonable paths planning could increase the network throughput trading off acceptable delays to achieve optimized network utility.

There are numerous case studies that require fine-grained traffic engineering, e.g., traffic filtering, load balancing, firewall, etc. Our work leverages the experiences learned from historical traffic data, which abundantly exist in current Internet, to achieve this target. The self-learning feature of the proposed model intends to make full use of existing experiences to discover implicit traffic trend.

6. Conclusion

In this paper, we have proposed a learning-based network-level forwarding method for path planning in the network with constrained conditions. We have sought to introduce well-defined deep learning model into network traffic engineering sphere by formulating the traffic forwarding problem as a sequence prediction problem. We have developed the learning-based model for traffic forwarding based on the sequence-to-sequence model enhanced by attention mechanism and beam search. Our proposed model has been implemented in the controller of an SDN architecture in Mininet emulator. Experiment results have shown the superiority of the proposed model in path planning in the network with constrained conditions.

Acknowledgement

This work is partially supported by the UK EPSRC project (Grant No.:EP/R030863/1)

References

[1] Z. Shu, J. Wan, J. Lin, S. Wang, D. Li, S. Rho, C. Yang, Traffic engineering in software-defined networking: Measurement and management, IEEE Access 4 (2016) 3246–3256.

[2] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, W. Chou, Research challenges for traffic engineering in software defined networks, IEEE Network 30 (3) (2016) 52–58.

[3] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, K. Mizutani, State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems, IEEE Communications Surveys Tutorials 19 (4) (2017) 2432–2455. doi:10.1109/COMST.2017.2707140.

[4] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, D. Yang, Experience-driven networking: A deep reinforcement learning based approach, in: Conference on Computer Communications, INFOCOM, IEEE, 2018.

[5] S. Chinchali, P. Hu, T. Chu, M. Sharma, M. Bansal, R. Misra, M. Pavone, K. Sachin, Cellular network traffic scheduling with deep reinforcement learning, in: National Conference on Artificial Intelligence (AAAI), 2018.

[6] J. Xiao, S. Chen, M. Sui, The strategy of path determination and traffic scheduling in private campus networks based on sdn, Peer-to-Peer Networking and Applications (2017) 1–10.

[7] S. B. H. Shah, Z. Chen, F. Yin, Open: Optimized path planning algorithm with energy efficiency and extending network-lifetime in wsn, Journal of computing and information technology 25 (1) (2017) 1–14.

[8] M. Arifuzzaman, O. A. Dobre, M. H. Ahmed, T. M. N. Ngatched, Joint routing and mac layer qos-aware protocol for wireless sensor networks, in: IEEE Global Communications Conference (GLOBECOM), 2016, pp. 1–6. doi:10.1109/GLOCOM.2016.7841933.

[9] Y. Niu, Y. Liu, Y. Li, X. Chen, Z. Zhong, Z. Han, Device-to-device communications enabled energy efficient multicast scheduling in mmwave small cells, IEEE Transactions on Communications 66 (3) (2018) 1093–1109. doi:10.1109/TCOMM.2017.2773529.

[10] E. Oh, B. Krishnamachari, Energy savings through dynamic base station switching in cellular wireless access networks, in: IEEE Global Telecommunications Conference GLOBECOM 2010, pp. 1–5. doi:10.1109/GLOCOM.2010.5683654.

[11] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, Openflow: enabling innovation in campus networks, ACM SIGCOMM Computer Communication Review 38 (2) (2008) 69–74.

[12] A. Lara, A. Kolasani, B. Ramamurthy, Network innovation using openflow: A survey, IEEE communications surveys & tutorials 16 (1) (2014) 493–512.

[13] E. Ahmed, I. Yaqoob, I. A. T. Hashem, I. Khan, A. I. A. Ahmed, M. Imran, A. V. Vasilakos, The role of big data analytics in internet of things, Computer Networks 129 (2017) 459–471.

[14] V. Marx, Biology: The big challenges of big data, Nature 498 (2013) 255. doi:10.1038/498255a.

[15] M. Taheriyan, C. A. Knoblock, P. Szekely, J. L. Ambite, Learning the semantics of structured data sources, Web Semantics: Science, Services and Agents on the World Wide Web 37 (2016) 152–169.

[16] A. Halevy, P. Norvig, F. Pereira, The unreasonable effectiveness of data, IEEE Intelligent Systems 24 (2) (2009) 8–12. doi:10.1109/MIS.2009.36.

[17] D. Crankshaw, P. Bailis, J. E. Gonzalez, H. Li, Z. Zhang, M. J. Franklin, A. Ghodsi, M. I. Jordan, The missing piece in complex analytics: Low latency, scalable model management and serving with velox, arXiv preprint arXiv:1409.3809.

[18] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436.

[19] R. Feynman, F. Vernon Jr., Sequence to sequence learning with neural networks, In Advances in neural information processing systems, NIPS (2014) 3104–3112.

[20] M. Huang, W. Liang, Z. Xu, W. Xu, S. Guo, Y. Xu, Dynamic routing for network throughput maximization in software-defined networks, in: IEEE INFOCOM, The 35th Annual IEEE International Conference on Computer Communications, 2016, pp. 1–9. doi:10.1109/INFOCOM.2016.7524613.

[21] A. Hari, U. Niesen, G. Wilfong, On the problem of optimal path encoding for software-defined networks, IEEE/ACM Transactions on Networking 25 (1) (2017) 189–198. doi:10.1109/TNET.2016.2571300.

[22] A. Azzouni, R. Boutaba, G. Pujolle, Neuroute: Predictive dynamic routing for software-defined networks, 13th International Conference on Network and Service Management (CNSM) (2017) 1–6.

[23] R. Srikant, The mathematics of internet congestion control, Springer Science & Business Media, 2012.

[24] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.

[25] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (1997) 17351780. doi:10.1162/neco.1997.9.8.1735.

[26] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, (2017) arXiv, http://arxiv.org/abs/1406.1078.

[27] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, In International Conference of Learning Representation, ICLR (2015).

[28] M. Luong, H. Pham, C. D. Manning, Effective approaches to attention-based neural machine translation, In Conference on Empirical Methods in Natural Language Processing, EMNLP (2015).

[29] G. Neubig, Neural machine translation and sequence-to-sequence models: A tutorial, (2017) arXiv, http://arxiv.org/abs/1703.01619.

[30] B. Lantz, B. Heller, N. McKeown, A network in a laptop: Rapid prototyping for software-defined networks, in: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, Hotnets-IX, ACM, New York, NY, USA, 2010, pp. 19:1–19:6. doi:10.1145/1868447.1868466.

[31] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, Tensorflow: A system for large-scale machine learning, in: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, OSDI'16, USENIX Association, Berkeley, CA, USA, 2016, pp. 265–283.

[32] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, (2014) arXiv preprint arXiv:1412.6980.

Biographies

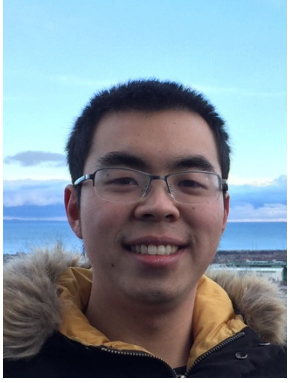
YUAN ZUO received his B.Sc. in Engineering from University of Electronic Science and Technology of China in 2012 and his M. Sc. in Engineering from National University of Defence Technology in 2014. He is currently a Ph.D. candidate in the College of Engineering, Mathematics and Physical Sciences at University of Exeter. His research interests include Machine learning, Neural Networks, Data Mining and Network traffic control.

YULEI WU received the B.Sc. degree (First Class Hons.) in Computer Science and the Ph.D. degree in Computing and Mathematics from the University of Bradford, U.K., in 2006 and 2010, respectively. He is currently a Lecturer in the Department of Computer Science with the University of Exeter, UK. He has published over 60 research papers in prestigious international journals and at reputable international conferences. His main research interests include Future Internet Architecture and Technologies, Smart and/or Active Network Management, Green Networking, Big Data for Networking, and Analytical Modeling and Performance Optimization.

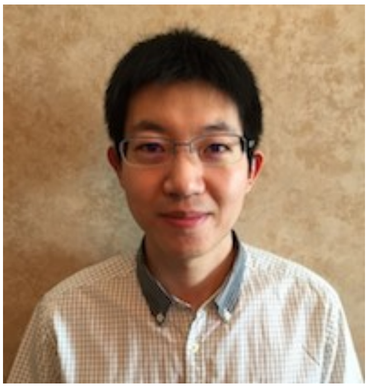
GEYONG MIN is currently the Chair Professor and Director of High Performance Computing and Networking (HPCN) Research Group at the University of Exeter, UK. He received the PhD degree in Computing Science from the University of Glasgow, UK, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. He joined the University of Bradford as a Lecturer in 2002, became a Senior Lecturer in 2005 and a Reader in 2007, and was promoted to a Professor in Computer Science in 2012. His main research interests include Next-Generation Internet, Analytical Modelling, Cloud Computing, and Big Data.

Photos

YUAN ZUO



YULEI WU



GEYONG MIN

Highlights

- A sequential model learns implicit paths from historical traffic experiences
- Attention mechanism captures correlations between source paths and target paths
- Beam search guarantees path connectivity by holding candidate paths
- A high testing accuracy implies the superiority of our proposal