Peer-to-Peer Networking and Applications (2022) 15:529–548
https://doi.org/10.1007/s12083-021-01262-8

Application-aware QoS routing in SDNs using machine learning techniques

Weichang Zheng[1] · Mingcong Yang[1] · Chenxiao Zhang[1] · Yu Zheng[1] · Yunyi Wu[1] · Yongbing Zhang[1] · Jie Li[2]

Received: 16 April 2021 / Accepted: 2 November 2021 / Published online: 11 November 2021

Abstract
Software Defined Networking has become an efficient and promising means for overcoming the limitations of traditional networks, e.g., by guaranteeing the corresponding Quality of Service (QoS) of various applications. Compared with the inherent distributed characteristics of the traditional network, SDN is logically centralized and can utilize machine learning techniques to keep track of transmission requirements of each application. In this research, we first develop an efficient data dimension reduction approach by considering the correlation coefficients between data items. We classify the traffic data into distinguished categories based on the QoS requirements by a supervised machine learning method. Then, we propose a QoS Aware Routing (QAR) algorithm according to the QoS requirements of each application that finds a path with either the minimum average link occupied times or the maximum average path residual capacity. The accuracy of machine learning model shows that our proposed dimension reduction approach is more effective than other data preprocessing methods, and the results of blocking probability indicate that our QAR algorithm outperforms significantly previous algorithms.

Keywords Software defined networking · Machine learning · Traffic classification · Quality of service routing

1 Introduction

The Internet traffic volume and the number of network applications nowadays are inexorably pushing the structural limitation of current IP networks. In recent years, Software Defined Networking (SDN) [4, 7] has been put forward as a feasible solution for overcoming these limitations. The new paradigm of SDN separates controlling and data forwarding functionalities of switches into the *control* and the *data* plane to get over the management difficulties in traditional IP networks. A central controller in SDN has the global information of the data forwarding switches in the network and the right to control all the devices. The controller makes the routing decisions dynamically for all the requests and performs the flow management, realizing the per-flow or application-aware QoS provisioning. In view of the SDN characteristics mentioned above, some algorithms for routing allocation have recently been implemented in SDN that take Quality of Service (QoS) metrics (e.g., routing latency and path bandwidth) into account [6, 23, 44].

There are many types of network applications, each of which may have different QoS requirements compared with others [12], and in the 5G networks the type of applications is expected to increase sharply [47]. According to [12], the

✉ Weichang Zheng
wchangzhengstu@gmail.com

Mingcong Yang
ymc6642664@gmail.com

Chenxiao Zhang
s2030111@s.tsukuba.ac.jp

Yu Zheng
s2020496@s.tsukuba.ac.jp

Yunyi Wu
s2030109@s.tsukuba.ac.jp

Yongbing Zhang
ybzhang@sk.tsukuba.ac.jp

Jie Li
lijiecs@sjtu.edu.cn

[1] Graduate School of Systems and Information Engineering, University of Tsukuba, 1 Chome-1-1 Tennodai, Tsukuba, Ibaraki 306-8577, Japan

[2] Department of Computer Science and Engineering, Shanghai Jiaotong University, 800 Dongchuan RD. Minhang District, Shanghai, China

latency requirement of a video streaming or phone-to-phone application should be no more than 150 milliseconds while requirements for a web-based application is generally more tolerable. Furthermore, to guarantee the quality level of the application, the channel bandwidth is also important. Therefore, routing decision should take both latency and bandwidth requirements of each traffic flow into account.

In this paper, we design a QoS Aware Routing (QAR) algorithm that considers two key QoS parameters: (1) the latency of data transmission for each application; and (2) the network bandwidth required by each application, in routing decisions. We use a set of real traffic history records to examine our proposed routing algorithm. However, a real dataset usually contains a large number of data items and many items may be highly correlated or completely ineffective. To reduce the data dimensions of the dataset, we propose an efficient approach to eliminate the correlated data items using Integer Linear Programming (ILP) technique.

In all, our proposed scheme in combination with DNN and QoS routing control is implemented based on SDN. The basic idea of it is to make the most suitable routing decision for each request with the consideration of QoS requirements. We apply the DNN model in SDN controller to do QoS classification dynamically for each request arriving at network. Then, the routing path of each request is generated through our proposed QoS Aware Routing (QAR) algorithm by SDN controller. Due to this architecture, as shown in Fig. 1, our proposed work is composed of the offline phase and the online phase.

In the offline phase, we use the labeled traffic dataset consisting of training data, validation data and testing data to train the DNN model and deploy it in the SDN controller. In the online phase, the SDN controller collects each arriving traffic request information, then the trained DNN model receives the information from controller and classifies the QoS requirements for each request. Finally, based on the QoS requirements, the SDN controller makes the routing decision and updates the network state in switches. The detailed offline and online phase will be introduced in Sect. 3.

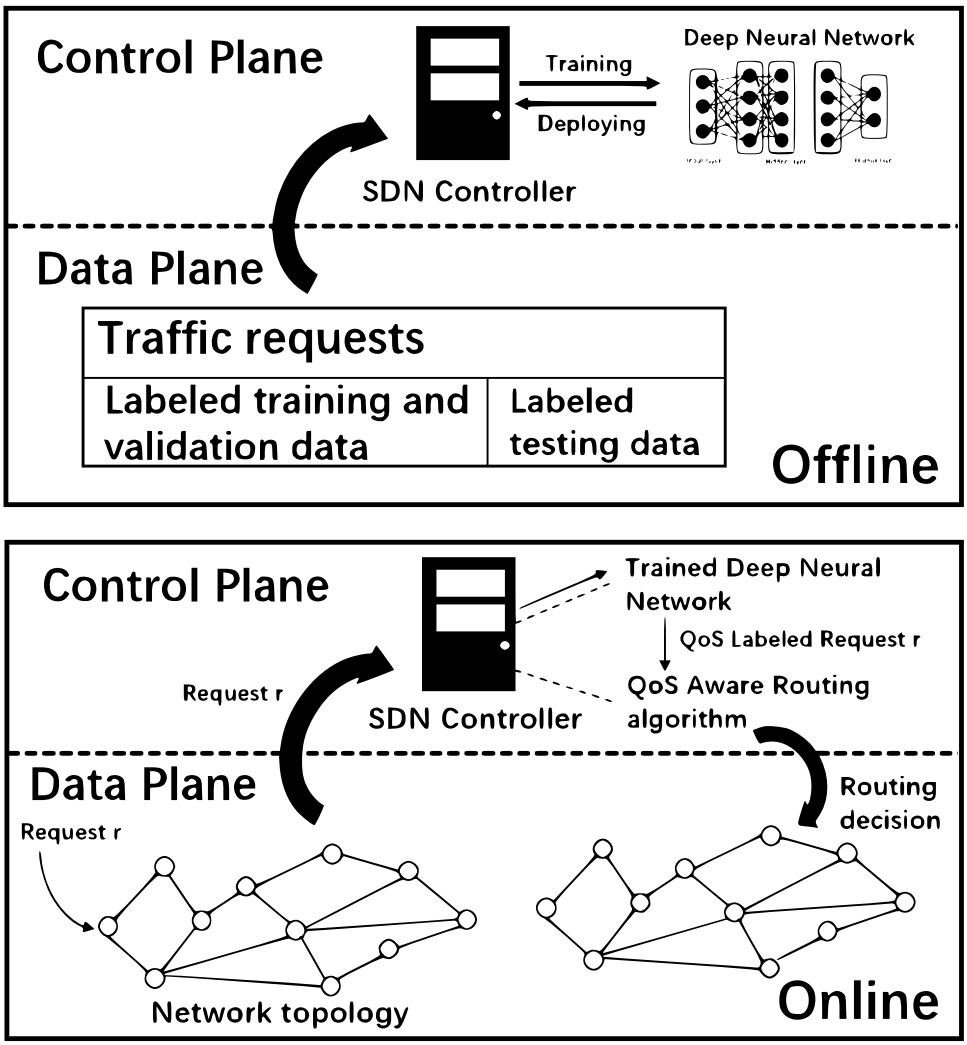


Fig. 1 The architecture and working phases of our proposed scheme

The main contributions in this paper are as follows. (1) We analyze a real traffic dataset using Deep Neural Network (DNN), which is a kind of supervised machine learning technique, so that the QoS requirements of each traffic flow can be clarified correctly. Before that, we remove the correlated data items using the ILP-based technique. The results obtained using this approach are significantly better than those used by the principal component analysis (PCA) approach [22], Least Absolute Shrinkage and Selection Operator (LASSO) [30], Random Forest (RF) [9] or previous researches which are widely used for data dimension reduction. (2) According to different flow request QoS requirements, we propose a QoS routing algorithm (QAR) that aims to find the most suitable path from the source to its destination node while minimizing number of times links are occupied or maximizing the average path residual bandwidth. The proposed algorithm yields better performance than other previous routing algorithms.

The remainder of this paper is organized as follows. Section 2 briefly introduces the background of SDN, QoS routing and ML-based traffic classification. In Sect. 3, we present our feature dimension reduction approach and a comparison of performance in machine learning with various feature reduction methods. Then we describe the routing model in SDN and the design of QoS Aware Routing (QAR) algorithm. In Sect. 4, we explain details of the simulation and the performance evaluation. Finally, in Sect. 5, we draw conclusions and future works of this paper.

2 Background and related work

2.1 Software Defined Networking (SDN)

To provide best-effort data transmission service, traditional IP network is constructed by deploying massive amounts of network devices (nodes) such as switches and other relay communication nodes [4, 7]. Each node operates independently and autonomously in the sense that not only the data packet forwarding but also the routing decision at each node. Various nodes may employ distinct operating systems that are incompatible with each other. In addition to adapt to complicated changing network requirements, network operators also need to transform protocols into machine configuration commands in different vertically inherited devices. This makes network monitoring, management and performance tuning quite complicated and error-prone.

SDN is split into three independent planes as shown in Fig. 2: *application plane*, *control plane* and *data plane*. Hence, the global monitoring of the network is separated from the forwarding mechanism so that the controller can centrally provide management with information of all the devices. The controller deployed in the control plane controls all the data forwarding devices (switches) and therefore can realize services such as security, QoS (Quality of Service), traffic engineering, etc. for the whole network [10]. It makes routing decisions and manages switches via an open protocol such as the Open-Flow [28]. In other words, SDN switches are controlled and programmed in the control plane, and the controller can keep monitoring at any time to guarantee the whole network's stability.

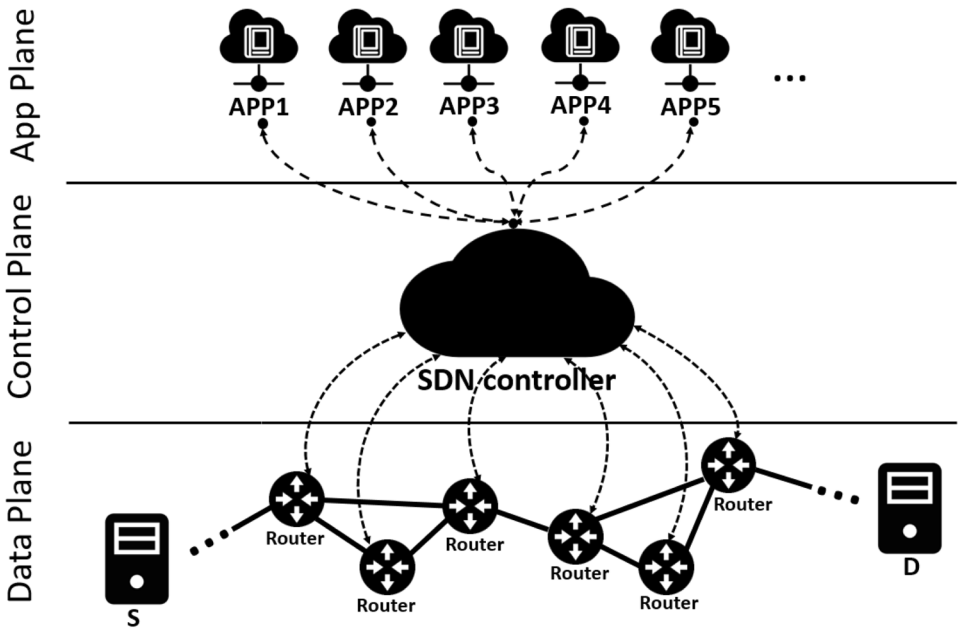


Fig. 2 The overall architecture of Software Defined Networking (SDN)

Since the controller can interact directly with switches in the data plane through the Open-Flow protocol, it collects a massive amount of network-related information, such as the network state, configuration data, packet information, and flow-granularity information. By analyzing the traffic data collected at the controller, better routing decisions can be made depending on the flow requirements.

2.2 QoS routing in SDN

The Quality of Service (QoS) for data transmission can be evaluated based on several parameters such as latency, bandwidth, packet loss, jitter (delay variation), etc [12, 15, 41]. Among them, transmission latency and bandwidth are usually considered as the two key performance measures for assessing the quality of data transmission. For this reason, this paper only considers the transmission latency and bandwidth when making routing decisions. Therefore, for each traffic flow, SDN seeks to guarantee the available bandwidth on the selected path and the transmission latency. In order to preserve resources, the SDN, therefore, takes into consideration different level of QoS requirements for traffic requests.

There are several routing algorithms [3, 25] that can be applied to SDN, in which a logically centralized controller is employed to manage switches and monitor network state information. After the suitable paths are determined, the routing decisions are installed at the switches along those selected paths. Akin and Korkmaz [3] proposed two different routing algorithms that, separately, take into consideration static and dynamic conditions in SDN. However, those algorithms do not take into account the QoS requirements of each traffic flow. Furthermore, Layeghy et al. [25] put forward a routing algorithm platform called software-defined constrained optimal routing (SCOR) platform. It consisted of three routing algorithms: the least cost path routing algorithm which was the same as the shortest path first (SPF) algorithm; the least cost path with link capacity constraints routing algorithm (SPF with CC), which should satisfy the bandwidth of each request; and the maximum residual capacity routing (MRCR) algorithm, which aimed to find the suitable routes for all the requests while maximizing the minimum residual link capacity in the paths. However, none of those three algorithms consider both the transmission latency and the bandwidth requirement at the same time.

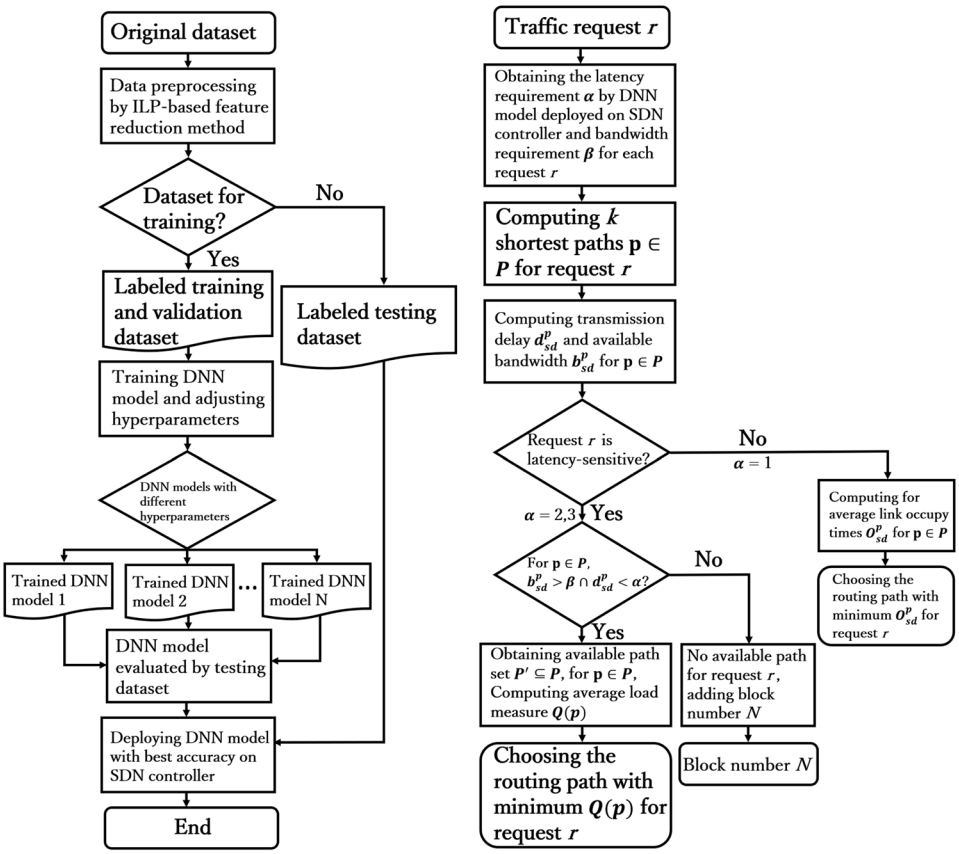
2.3 Machine learning for traffic classification

Various machine learning techniques [11, 24, 27] have been developed to help analyze huge amount of data. Unsupervised, supervised, semi-supervised and reinforcement learning are the four most basic machine learning algorithms. Nevertheless, the inherently distributed nature of traditional IP network architecture makes it generally difficult to analyze network performance by applying machine learning techniques. Central the concept of SDN is that, the controller collects all information from switches and therefore has the global overview of the network. This enables better configuration of the machine learning techniques for traffic analysis, routing decision, network management, etc [23]. The controller becomes more efficient in data analysis, topology optimization, traffic allocation and other network services. In this way, the controller is able to optimize the whole network resource allocation with corresponding constraints. The machine learning techniques have been applied to tasks in SDN in a number of recent studies.

Amaral et al. [6] used supervised learning in SDN for application-aware traffic classification. The number of applications they tested is limited to eight, an unrealistically small number compared to real networks. Chhabra and Kiran [13] used supervised learning for traffic size classification (mice and elephant flow). However, as mice flow accounts for nearly 90 percent of the total traffic flow [5], the value of this approach to discuss classification accuracy is unclear. Xiao et al. [43] used spectral clustering for service-aware classification. Limitations of this approach are that (1) the selection of clustering parameters is sensitive and strict, and (2) the data requires normalization because of the large number of features [16] and possible different numerical dimensions. Erman et al. [18] aimed at traffic classification using semi-supervised learning method and they first used K-Means clustering for training data partition and then made the clusters mapping to different flow byte types. However, it is preferable to classify the byte sizes of traffic flows into several levels instead treating the value of flow bytes as the label. Yamansavascilar et al. [45] evaluated the traffic classification by four different algorithms and focused on popular applications classification which is readily modifiable due to the explosive growth of Internet traffic. Another previous work by Erman et al. [17] used clustering algorithms to do traffic application classification. The clustering algorithms such as K-Means are very strict in parameter setting. Zhang et al. [49] focused on the robustness classification through the combination of supervised and unsupervised learning and proposed a RTC scheme to identify the zero-day applications that was unknown in advance. Zhang et.al. [50] also proposed a correlation information-based traffic classification method which could perform well even with few data samples.

3 Traffic classification and QoS routing

In this paper, as shown in Fig. 3 we first classify various traffic flows into distinct levels depending on their quality of service, e.g., latency requirement. We use the historical Internet traffic flows and DNN model for traffic classification. Then,



The flowchart of traffic classification. The flowchart of QoS routing decision.

Fig. 3 The diagram of traffic classification and routing decision

we propose a QoS-aware routing approach to determine the best route for each newly routing request.

3.1 Traffic classification

3.1.1 Data dimension reduction and selection

In this paper, we use a dataset of real network traffic histories [2] which includes 75 applications and 3.57M transmission records, each of which contains 87 items (features). Here, we let $F = \{1, 2, \ldots, 87\}$ to denote the set of features.

In order to speed up the data analysis, a dimension reduction method such as the principal component analysis (PCA) [22] is usually used. However, the PCA based approach has limitations with highly nonlinear data [14] and unfortunately the dateset used in this paper is not linear. LASSO method [30] imposes a constraint on the sum of the absolute values of the model parameters. By penalizing the coefficients of the regression variables, some of them shrink to zero, which can be used as a kind of feature selection method. In the process of LASSO modeling, the tuning parameter λ assumes a great importance to control the strength of the penalty [20]. The larger parameter λ will result in more numbers of feature coefficient reducing to zero. However, setting of the parameter λ is strict to LASSO modeling. If the dimension of features p exceeds the observations n, LASSO can at most select $n \ll p$ features [38]. In addition, the real world datasets usually have groups of correlated features in general, LASSO selects only one feature from each group arbitrarily in nature. The grouped variables which are highly correlated with each other only can be selected one from each group by LASSO but ignores the others. An ensemble learning algorithm based on decision tree called Random Forest (RF) can also be used for feature reduction [37]. RF selects features through comparing the contribution of each feature to each tree in the random forest called importance. The features whose importance are greater than a certain threshold γ are selected. However, how to choose the threshold γ has no definite standard. The setting of number of decision trees has an effect on the distribution of features' importance.

To overcome the limitations of PCA, LASSO and RF, we propose an effective data reduction method in which the correlation coefficients between features should be greater than a predefined value δ. To this end, we formulate an integer linear programming (ILP) problem, and attempt to maximize the number of features so that the correlation coefficient r_{ij} between two features i and j ($i, j \in F, i \neq j$) is less than δ as follows.

$$\max \sum_{i \in F} x_i \tag{1}$$

subject to

$$r_{ij} \leq M(2 - x_i - x_j) + \delta, \quad \forall i, j \in F, i \neq j \tag{2}$$

$$r_{ij} \geq -M(2 - x_i - x_j) - \delta, \quad \forall i, j \in F, i \neq j \tag{3}$$

$$x_i, x_j \in \{0, 1\} \tag{4}$$

where M is a large positive number greater than any value used in problem (1), and x_i, x_j denote indicate variables for features $i, j \in F, i \neq j$ and if $x_i = 1$, feature i is selected and $x_i = 0$, otherwise. Constraint (2) guarantees that if two features have a positive relation, their correlation coefficient should be less than δ. On the other hand, constraint (3) guarantees that if two features have negative relation, their correlation efficient should be greater than $-\delta$. In this paper we selected δ as 0.3. The optimization problem (1) can be solved by using a tool for solving linear programming problem like the Gurobi Optimizer [1]. We use Gurobi Optimizer 8.1.1 by python 3.7 on Windows 10, Intel(R) Core(TM) i7-8700k CPU @ 3.70GHz to solve problem (1). The time complexity for solving an ILP problem is governed by the numbers of variables and constraints. The computing time complexity of problem (1) is $O(n^2)$. We obtain 24 features from the original 87 features. Note that our proposed method can be applied to any other dataset for dimension reduction.

In order to make our data analysis meaningful, we select 18 applications whose records are more than 500 to be classified according to their QoS requirements as shown in Fig. 4. Furthermore, we choose 90,000 records from the dataset as our training set, 10,000 records constituted the validation dataset and 10,000 records for testing in machine learning. Here, for the selected 18 applications, we first choose all the records of the applications whose records are less than 10,000. Then, we randomly choose the equal 8936 records of each remaining application.

As [12] indicated, some different applications may have the same QoS requirements such as the transmission latency and the required bandwidth. In this paper, we focus simply on the latency requirement of each traffic flow as the QoS measure to be classified (bandwidth of each flow is already definite before routing decision). According to [12], all the applications can be classified into three classes according the latency requirement, that is, class 1 : *not sensitive*; class 2: *weakly sensitive* (< 250 milliseconds); class 3: *strongly sensitive* (< 100 milliseconds). For the 100,000 training data and validation records, we assign each traffic flow record, i.e., each application, traffic class label and latency label as shown in Table 1. For the 10,000 testing data records, the information is given about the application name, traffic class and latency requirement, which is used for accuracy testing.

3.1.2 Traffic classification using machine learning technique

We use a general supervised learning algorithm [11, 24] called the Deep Neural Network (DNN) for traffic classification in this paper. Compared with other supervised learning algorithms, the DNN can be applied to various types and distributions of datasets, and it can provide better performance when dealing with large-scale datasets. Table 2 reveals the traffic classification accuracy and the model training time via different supervised learning algorithms.

We see from Table 2 that both the training and testing accuracy of the DNN outperform other methods. Random Forest (RF) with different numbers of decision trees (estimators) behave well but worse than DNN in classification accuracy. What's more, increasing the number of decision trees has little effect on accuracy. However, the RF is easier

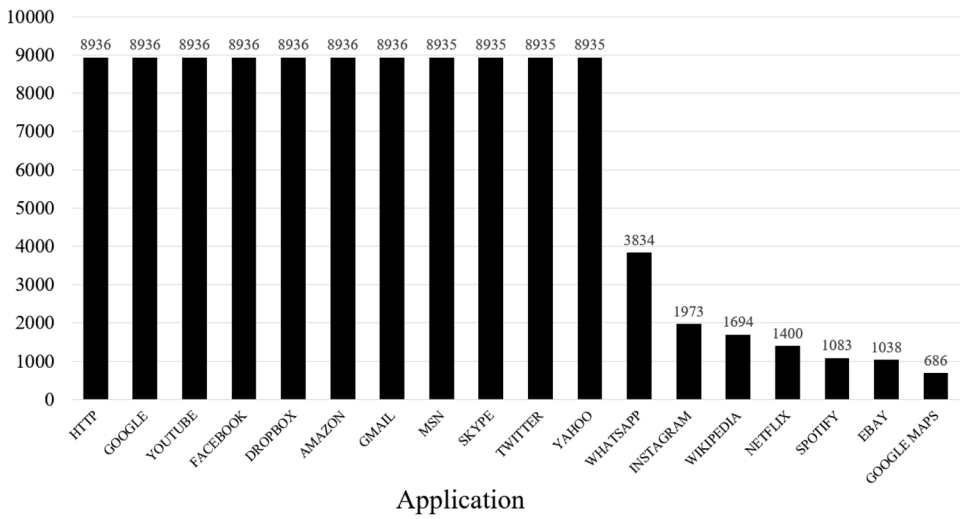


Fig. 4 The chosen number of applications for machine learning

Table 1 QoS class definitions of traffic flows

Application	Traffic class	latency req.
YOUTUBE	VIDEO STREAMING	strongly sensitive
NETFLIX	VIDEO STREAMING	strongly sensitive
SPOTIPY	VIDEO STREAMING	strongly sensitive
WHATSAPP	VOIP	strongly sensitive
SKYPE	VOIP	strongly sensitive
FACEBOOK	RELAY CHAT	weakly sensitive
INSTAGRAM	RELAY CHAT	weakly sensitive
TWITTER	RELAY CHAT	weakly sensitive
MSN	RELAY CHAT	weakly sensitive
GOOGLE-MAPS	TELEMETRY	weakly sensitive
HTTP	WEB	not sensitive
GOOGLE	WEB	not sensitive
AMAZON	WEB	not sensitive
WIKIPEDIA	WEB	not sensitive
YAHOO	WEB	not sensitive
EBAY	WEB	not sensitive
GMAIL	MAIL	not sensitive
DROPBOX	FTP,BULK	not sensitive

to over fit in classification problems with large-noise datasets. Therefore, the datasets used for training RF model need more strict preprocessing [29]. Another supervised learning method called Naive Bayes (NB) has bad performance on this classification. No matter using Gaussian, Multinomial or Bernoulli model, they behave no well in classification accuracy. Because the classification by NB is through the prior and data itself to determine the posterior probability, leading to a certain error rate during the classification decision [35]. The Support Vector Machine (SVM) has low classification accuracy in our multi classification problem. SVM is mainly used in binary classification so that we solve our multi classification problem by the combination of multiple binary SVMs. However, SVM is difficult to implement for large-scale training samples and it is sensitive to the selection of parameters and kernel function [8, 32]. Except for the classification accuracy, we also compared the model training time. Due to our offline classification model training, the training time is not the main necessary factor for evaluation of different supervised learning algorithms. Moreover, DNN does not take much time for training compared with other methods. Nevertheless, due to the complexity of multi classification SVM, its training time is the longest. In summary, we use the DNN model for traffic classification in this paper.

Table 3 Parameters for Deep Neural Network model

Item	Parameter setting
Input layer	Num. of neuron units: 24
	Activation function: Relu
	Drop out rate: 0.1
Hidden layer	Num. of layers: 10
	Num. of neuron units: 200
	Activation function: Relu
	Drop out rate: 0.1
Onput layer	Num. of neuron units: 3
	Activation function: Softmax
Loss function	Cross entropy
Batch size	1,000
Epoch	100

The DNN model and its corresponding parameters are defined as shown in Table 3. The samples of traffic records are denoted by x while the labels are denoted by $y = (N, \alpha)$ where N and α are latency requirement classes ($N = \{1, 2, 3\}$) and required latency respectively. Therefore, by training the DNN model, we can estimate the latency requirement (N and α) of each arriving traffic flow.

For the learning process, we use the training dataset with 90,000 samples to train DNN model and fit the weights of classifier, and each real traffic record has different number of features according to different feature reduction methods. For example, the dataset preprocessed by our proposed ILP-based feature reduction method has 24 features, that is to say, the size of training data matrix is $90{,}000 \times 24$ and the size of training label matrix (vector) is $90{,}000 \times 1$. Then the validation dataset with 10,000 labeled records is used to evaluate the model and adjust hyperparameters, the testing dataset with 10,000 labeled samples is applied for final evaluation

Table 2 Comparison of classification accuracy and model training time

Method	Train acc.	Test acc.	Train time (s)
Deep Neural Network	0.9863	0.9676	177.6
Random Forest(estimators num. = 10)	–	0.9159	1.8
Random Forest(estimators num. = 100)	–	0.9517	17.2
Random Forest(estimators num. = 1000)	–	0.9564	175.1
Naive Bayes (Gaussian)	0.3693	0.2631	0.2
Naive Bayes (Multinomial)	0.5549	0.1525	0.1
Naive Bayes (Bernoulli)	0.5340	0.2078	0.1
SVM	0.4684	0.4352	>10hrs

[34]. Therefore, the learning process of DNN model including training and validation is shown in Fig. 5.

The training and the testing accuracy of our approach compared with different approaches including: one previous work proposed, three PCA-based, two LASSO-based and one Random Forest-based are illustrated in Table 4. The accuracy on the training data and the testing data by our proposed feature preprocessing method are 0.9958 and 0.9786, respectively. On the other hand, if the data features are reduced by using PCA, the accuracy on the training and the testing data can be 0.7327 and 0.6399 in the best case, respectively, which is far worse than our method. What's more, whether the dataset is preprocessed by LASSO-based or Random Forest-based feature reduction method, the classification accuracy with those datasets behave better than PCA-based methods, but they are not inferior to our proposed ILP-based method.

Different feature reduction methods mentioned in Table 4 and their corresponding selected features are as follows:

- *This research*: 24 features selected by our proposed ILP-based feature reduction approach are as follows: total quantity of forward packets, minimum value of backward packets length, mean value of backward packets length, number of bytes per second, minimum bi-directional inter-arrival time, mean forward inter-arrival time, number of forward packets per second, number of backward packets per second, minimum packets length, times FIN flag is marked, times SYN flag is marked, times RST flag is marked, times URG flag is marked, download and upload ratio, average forward segment size, average packets in a backward subflow, total bytes sent in the forward initial window, total bytes sent in the backward initial window, minimum forward segment size, standard deviation active time, minimum active time, standard deviation idle time, minimum idle time, the layer 7 protocol code number.
- *Previous work*: 9 commonly traffic flow features are used in [6, 13] as follows: source port number, source IP address, protocol used, flow duration, bytes of data transferred, packets of data transferred, mean value of the inter-arrival time, destination port number, destination IP address.
- *PCA1*: 23 items (components) are selected by using PCA approach on condition that $\eta = \sum_i r_i > 99.99\%$ where r_i denotes the contribution ratio by principal component i.
- *PCA2*: 70 items are selected by using PCA approach on condition that the eigenvalue λ_i of component i is greater than 0.
- *PCA3*: 5 items are selected by using PCA approach on condition that $\eta = \sum_i r_i > 95\%$ where r_i denotes the contribution ratio by principal component i.
- *LASSO1*: 10 features selected by using LASSO approach with the threshold of tuning parameter $\lambda = 0.001$ are as follows: times pushing flag is marked, times FIN flag is

Table 4 Comparison of classification accuracy via different feature reduction methods

Method	Train accuracy	Test accuracy
This research	0.9958	0.9786
Previous work [6, 13]	0.5378	0.4586
PCA1 ($\eta > 99.99\%$)	0.6806	0.3467
PCA2 ($\lambda_i > 0$)	0.7327	0.6399
PCA3 ($\eta > 95\%$)	0.6128	0.2752
LASSO1 ($\lambda = 0.001$)	0.7153	0.4183
LASSO2 ($\lambda = 0.0001$)	0.9566	0.9292
Random Forest ($\gamma = 0.01$)	0.9635	0.9399

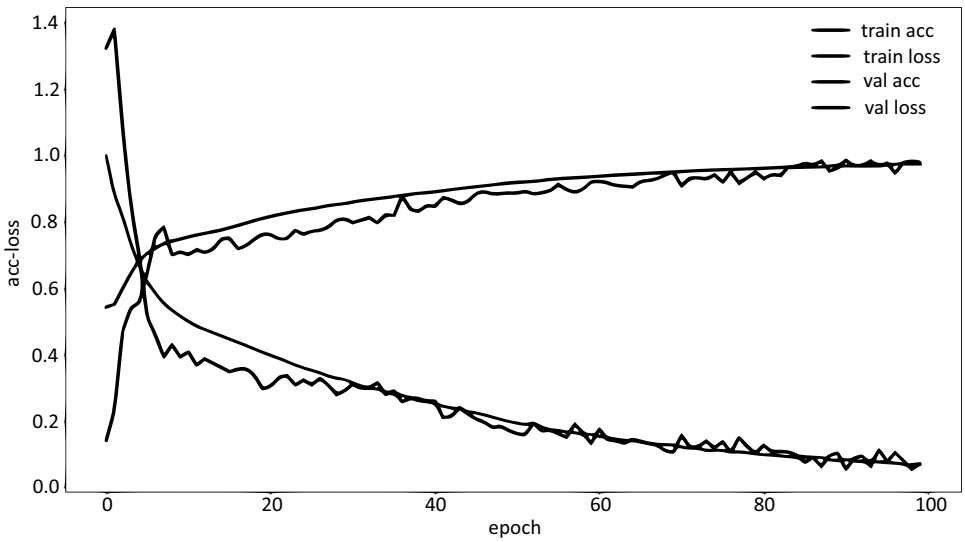


Fig. 5 The learning process of DNN with training and validation datasets

marked, times SYN flag is marked, times PSH flag is marked, times ACK flag is marked, times URG flag is marked, times ECE flag is marked, download and upload ratio, minimum forward segment size, the layer 7 protocol code number.

- *LASSO2*: 18 features selected by using LASSO approach with the threshold of tuning parameter $\lambda = 0.0001$ are as follows: minimum value of forward packets length, mean value of forward packets length, standard deviation value of forward packets length, maximum value of backward packets length, minimum value of backward packets length, mean value of backward packets length , times PSH flag is marked, minimum packets length, maximum packets length, times FIN flag is marked, times SYN flag is marked, times PSH flag is marked, times ACK flag is marked, times URG flag is marked, times ECE flag is marked, download and upload ratio, minimum forward segment size, the layer 7 protocol code number.
- *Random Forest*: 31 features selected by using Random Forest approach with the threshold of features importance $\gamma = 0.01$ ($\gamma = 0.01$ is better than other γ values) are as follows: source port number, destination port number, total flow duration, total quantity of backward flow bytes, maximum value of forward packets length, mean value of forward packets length, standard deviation value of forward packets length, maximum value of backward packets length, mean value of backward packets length, standard deviation value of backward packets length, number of forward packets per second, mean bi-directional inter-arrival time, standard deviation bi-directional inter-arrival time, maximum bi-directional inter-arrival time, minimum bi-directional inter-arrival time, total forward inter-arrival time, mean forward inter-arrival time, maximum forward inter-arrival time, total backward inter-arrival time, maximum backward inter-arrival time, number of forward packets per second, number of backward packets per second, maximum packets length, standard deviation packets length, average packet size, average forward segment size, average backward segment size, average bytes in a forward subflow,total bytes sent in the forward initial window, total bytes sent in the backward initial window, the layer 7 protocol code number.

With the comparison of traffic classification performance by DNN using different kinds of feature dimension reduction methods, we can obtain from Table 4 that, the accuracy of classification by our ILP-selected 24 features performs better than any other methods. What's more, when we use the 9 commonly used features by previous works, which means the features we used is most representative of the flow itself generally. However, the result is not as good as we expect because that too few features are selected, and some of them like Source IP and Destination IP are highly correlated which has a negative impact on classification.

In addition, we simulate the classification by Principal Component Analysis (PCA) technique. As shown in Table 4, by different conditions of dimension reduction, all the 3 methods of PCA perform not as well as our proposed method. With the analysis of PCA results, the limitations of PCA which cause the bad efforts on classification experiments are as follows. Firstly, the assumption of PCA is that the inter-class variance of data we use is larger than the inter-class variance, that means the principal components with highest variance will also contain the most information to do classification. However, the separation of our dataset is not on the highest variance. Secondly, as mentioned before, PCA actually is an unsupervised learning technique, which has a good effect on the dataset with linear correlation between features but not nonlinear situation [36].

In order to make a comparison, we also use LASSO as a feature reduction method to evaluate the performance. With different tuning parameter λ setting, various sets of features are selected by LASSO regression. According to the Table 4, the LASSO performs better than PCA and previous work but worse than our proposed method. LASSO can improve the prediction accuracy because that the feature coefficients shrinking reduces variance without the substantial increase of the bias [20]. However, as mentioned before, LASSO will select only one feature arbitrarily from a group of correlated features, and eliminate other features, yet we can not know whether those features to be eliminated are correlated with other groups or not. Moreover, LASSO is not suitable for the case that the number of features far outweigh the number of observations. The determination of parameter λ is also troubling [30, 31].

The nonlinear property of Random Forest makes it superior to linear algorithm, which makes it a good choice for traffic classification. However, the uncertainty of dataset and the non-inferability of random forest will affect the classification. The prediction range of a RF is limited by the label scale in the training data. The classification by RF behaves problematic when the distribution of training and prediction inputs are different [33].

In summary, those previous different feature reduction methods need to be associated with the actual dataset distribution, and results show that the consideration of correlation coefficient among features is necessary and has positive impact on feature reduction works.

3.2 QoS routing

3.2.1 Model description

In this section, we describe our proposed QoS Aware Routing (QAR) algorithm that determines the data transmission route

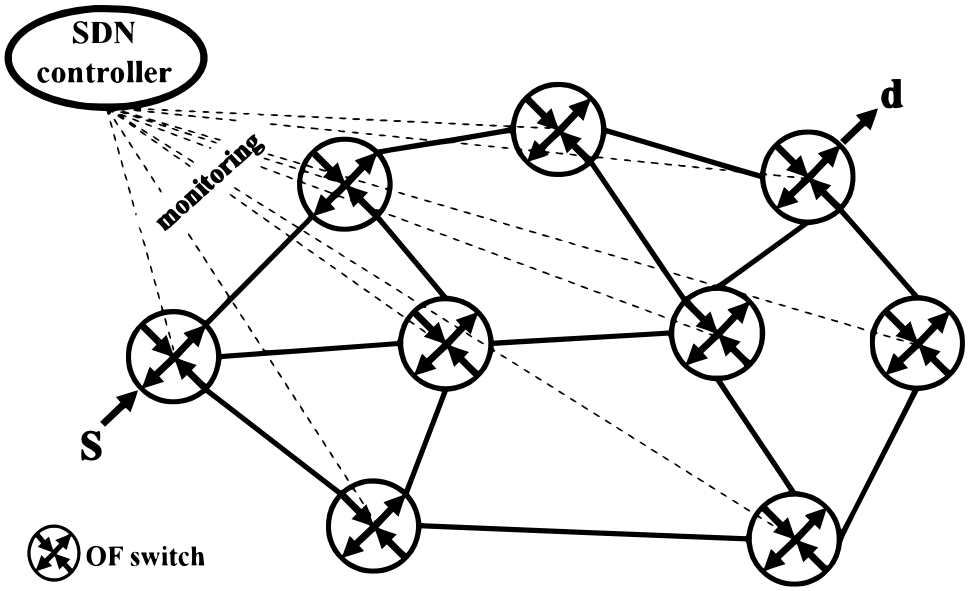


Fig. 6 A SDN model

for each traffic request flow starting from source terminal s to destination terminal d as shown in Fig. 6. All the routing requests arriving at nodes are sent to the central controller for making routing decisions. The proposed routing algorithm is essentially dynamic in the sense that the routing decisions are based on a short time period T previously from the current instant. The capacity of the link between node i and j, denoted by l_{ij}, is assumed to be equal to others and is denoted by c_{ij}.

Transmission latency of link l_{ij} is denoted by d_{ij}. In order to estimate the propagation latency t_p via nodes i and j, the controller sends estimation packets to itself via nodes i, j and vice versa as shown in Fig. 7. Also, the transmission latency d_{ij} is related to the Round Trip Time (RTT) for traffic request from controller to each switch (node) t_c and switch measuring time t_m. In all, we can calculate d_{ij} as follows:

$$d_{ij} = t_p + t_c + t_m, \tag{5}$$

Let the flow size sent from node i to j in T be f_{ij}, then we define the load of link l_{ij}, denoted by w_{ij}, as follows.

$$w_{ij} = \frac{f_{ij}}{T}. \tag{6}$$

Therefore, the available bandwidth of link l_{ij}, denoted by b_{ij}, is given by

$$b_{ij} = c_{ij} - w_{ij}. \tag{7}$$

For a source-destination node pair (s-d), let P_{sd} represent the available paths set of sd node pair and p ($p \in P_{sd}$) denotes a path from node s to d. Then, we can calculate the

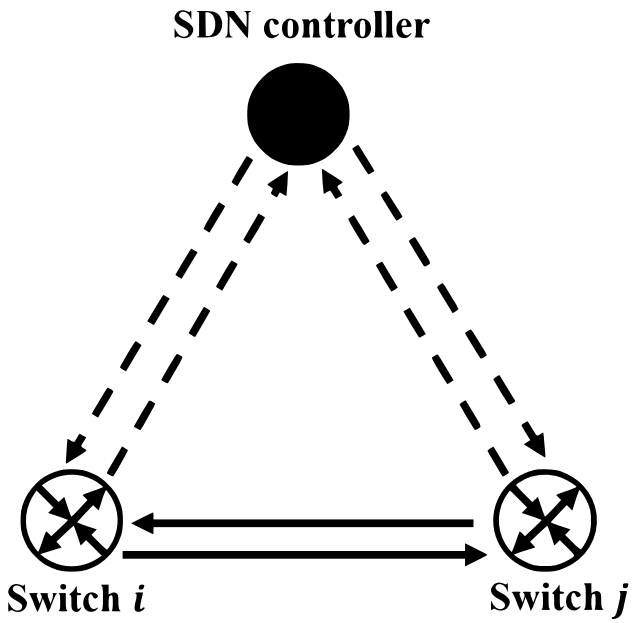


Fig. 7 Estimation of a link delay

transmission delay through path p ($p \in P_{sd}$), denoted by d^p_{sd} as follows.

$$d^p_{sd} = \sum_{l_{ij} \in p,\ p \in P_{sd}} d_{ij}. \tag{8}$$

Here, we define the available bandwidth of path p ($p \in P_{sd}$), denoted by b^p_{sd}, as follows.

$$b^p_{sd} = \min_{l_{ij} \in p,\ p \in P_{sd}} (C_{ij} - w_{ij}). \tag{9}$$

Table 5 Notation used in this paper

Symbol	Meaning
r_{ij}	correlation coefficient between data items (features) i and j
x_i	indicate variable showing whether feature i is selected for machine learning. $x_i = 1$, if feature i is selected and $x_i = 0$, otherwise
δ	threshold value of correlation coefficient between features
(s, d)	source and destination pair of request
α	latency requirement of request
β	bandwidth requirement of request
t_r	arrival time of request r
h_r	holding time of request r
α_r	latency requirement of request r
β_r	bandwidth requirement of request r
l_{ij}	link between nodes i and j
d_{ij}	link transmission latency of link l_{ij}
C_{ij}	link capacity of link l_{ij}
w_{ij}	load of link l_{ij}
$o_{ij}^{t_r}$	occupied times of link l_{ij} at arrival time of request r t_r
b_{ij}	available bandwidth of link l_{ij}
P_{sd}	set of paths of source-destination (s-d) pairs
d_{sd}^p	transmission latency of path $p \in P_{sd}$
b_{sd}^p	available bandwidth of path $p \in P_{sd}$
o_{sd}^p	average link occupied times of path $p \in P_{sd}$
n_{sd}^p	link number of path $p \in P_{sd}$

Since the requests arrive at each node at different times, and we set t_r represents the arrival time of request r. We define the occupied times as the times link being occupied by the requests still in transmission under current network state. If the request is released, the number of link occupied times will be reduced accordingly. Hence, the occupied times of link l_{ij} before request r arrives at time t can be computed by controller is denoted by $o_{ij}^{t_r}$.

In all, the used notation in this research is revealed in Table 5.

3.2.2 Proposed heuristic algorithm

The proposed heuristic algorithm attempts to maximize the number of transmission requests as many as possible and is described as follows.

1. For each transmission request with a (s, d) pair, we classify its traffic class and latency requirement using the machine learning approach. We can obtain the whole QoS requirement (N, α, β) for each request where $N = \{1, 2, 3\}$ denotes three types of latency requirement for the request. α is the concrete value of transmission latency requirement if sensitive, and β is the bandwidth requirement.
2. For the (s, d) pair of each request, we use Yen's K-shortest path algorithm [48] to find the k shortest paths, denoted by P_{sd}^K. By judging whether the request is sensitive to latency requirement or not, we propose routing schemes respectively.
3. Since the request r arrives at time t_r , if it is not sensitive to transmission latency, we then compute the average link occupied times of k paths selected by Yen's algorithm, denoted by o_{sd}^p which is defined as follows:

$$o_{sd}^p = \frac{\sum_{l_{ij} \in p_{sd}} o_{ij}^{t_r}}{n_{sd}^p} \tag{10}$$

Then we choose the minimum average link occupied times path with the satisfied available bandwidth b_{sd}^p, if there is no enough available bandwidth in chosen k paths, the request is blocked.
4. If the request r arrived at time t_r is sensitive, that means the latency requirement α equals to 100 milliseconds or 250 milliseconds. We define the average load measure for path p, denoted by $Q(p)$. Then we choose the minimum load measure path with the satisfied available bandwidth b_{sd}^p and transmission delay d_{sd}^p. If no such path can be found, we block the request.

$$Q(p) = \frac{C_{ij} - b_{sd}^p + \beta}{C_{ij}} \cdot \text{hop}\,(p). \tag{11}$$

The detail procedures are given in Algorithm 1.

Algorithm 1: QoS aware routing(QAR) algorithm

Input : (N_r, α_r): Request info obtained by machine learning;
β_r: Request bandwidth requirement;
P_{sd}^K: Paths obtained by K-shortest path algorithm.
Output: The routing path p for each request.

```
for p ∈ P_sd^K do
    Compute transmission delay d_sd^p;
    Compute available bandwidth b_sd^p;
    if request r is not sensitive to transmission latency then
        Compute average link occupied times o_sd^p;
        if b_sd^p < β_r then
            P_sd^K = P_sd^K \ p
        end
        if P_sd^K is empty then
            Block the request r
        else
            Selected path p such that p = arg min_{p∈P_sd^K} o_sd^p
        end
    else
        Compute the load measure Q(p);
        if b_sd^p < β_r then
            P_sd^K = P_sd^K \ p
        end
        if d_sd^p > α_r then
            P_sd^K = P_sd^K \ p
        end
        if P_sd^K is empty then
            Block the request r
        else
            Selected path p such that p = arg min_{p∈P_sd^K} Q(p)
        end
    end
end
```

4 Simulation experiments and performance evaluation

The parameter index we use to measure the network performance including the blocking probability and the average throughput. As shown in Fig. 8, simulation experiments are conducted in the European Optical Network (EON) topology [39] (28 nodes and 68 directed links) and the American ATT Backbone Network (AABN) topology [26] (27 nodes and 74 directed links) via the four routing algorithms mentioned above. The total quantity of requests used in simulation is 100,000 which are from the traffic dataset we used for classification. Moreover, the assumptions adopted in simulation experiments are as follows:

- Owing to the current Internet traffic asymmetry, each request with a specific *sd* node pair is assumed without regard to its directionality. That is to say, the amount of network links should be counted double.
- The capacity of links in the simulation networks is identical whose volume is generated from the range of 60 Gbps to 190 Gbps.
- According to the Electrical White paper Latency in optical fiber systems, the speed of light in optical fiber is 67% of c, so the propagation latency of link equals to:

$$v_p = \frac{1}{(3 \times 10^8 \times 0.67)} = 5\mu s/km = 0.005ms/km. \quad (12)$$

- The measuring latency of switches in the simulation network are all identical [42], and the value is 1∼5 milliseconds [21]. The latency (Round Trip Time) between switches and OpenFlow controller equals to 6∼8 milliseconds [40].
- Suppose that the requests arrive at each node in turn in the light of a Poisson process with the average arrival rate λ. The average duration of requests follows a negative exponential distribution with the holding coefficient μ [46]. Moreover, traffic volume of each request flow is generated based on Pareto distribution [19], ranging from 1 Gbps to 21 Gbps. The total collection of data transferred per request flow is the product of traffic volume and average duration. Therefore, the more network bandwidth allocated to each request during transmission with the larger holding coefficient μ.

The holding coefficient μ in simulation is set as 20, 30 and 50. By comparison with other previous routing algorithms: Shortest Path First (SPF) algorithm, Shortest Path First with Capacity Constraint (SPF with CC) algorithm and Maximum Residual Capacity Routing (MRCR) algorithm, the performance metrics are as follows:

(1) the blocking probability: $\eta = \frac{n}{N}$, here n is blocking request number, N is total request number.
(2) the average throughput: $\omega = \frac{\sum_{r=1}^{N} \beta_r \times h_r}{T}$, here h_r is the holding time of request r, T is the whole simulation time.

Figure 9 shows the blocking probability on the American ATT Backbone Network (AABN) for different holding coefficients and routing algorithms against the size of link capacity. As shown in Fig. 9, for each of the four routing algorithms, the blocking probabilities decreases when link capacity is increased from 60 Gbps to 190 Gbps. Indeed, when the holding coefficient μ is set at a general level (e.g., $\mu = 20, 30$), no matter how large the link capacity size is, our proposed QoS Aware Routing (QAR) algorithm performs better in comparison to the other three routing algorithms. However, if μ equals 50, the total transmission data carried by the requests is too heavy a load for the network.

As indicated in Fig. 9, when link capacity is small, the blocking probability is too high. This is because as the high value for μ equates to a long duration for each request and link capacity is insufficient. If link capacity is increased, the blocking probability will drop to an acceptable level, and our QAR algorithm has better performance. In addition, it should be mentioned that when link capacity increases to larger values, the blocking probabilities almost all reduce to zero. In this case, link capacity is enough to meet nearly any request requirements so that the impact of transmission latency on routing decisions becomes negligible.

From Fig. 10, we can see the blocking probability of the algorithms on the European Optical Network (EON) under the same conditions as results in Fig. 9. When link capacity is limited to small values, the blocking probability on the EON are higher than on the AABN with the same link capacity and holding coefficient settings. That is because the intermediate links in the EON are more frequently-used so that the load on them is always at a high level. The results in Fig. 10 reveal that our proposed QAR algorithm outperforms others, and the tendency for each group of curves with various μ is basically the same as those in Fig. 9.

Figures 11 and 12 show the relationship between average throughput and link capacity under different routing algorithms on the AABN and EON, respectively. As before, varying the μ setting for holding coefficients (here $\mu = 20, 30, 50$) results in different average network throughput. The average throughput of the network is defined as the amount of transferred traffic data per second within the entire simulation period of time. As revealed by these two figures, the average throughput is positively correlated with link capacity. Obviously, when we enlarge the

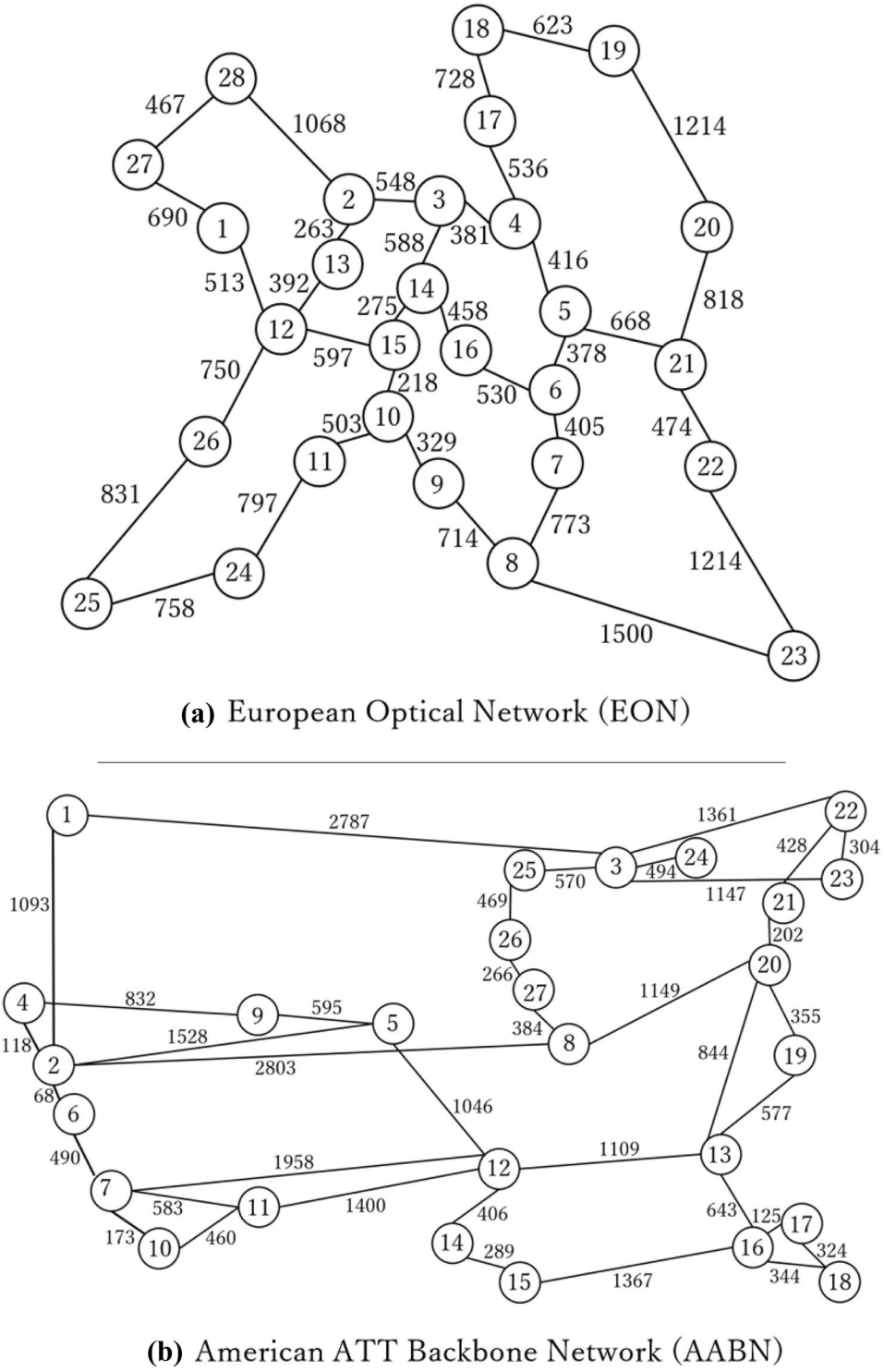


Fig. 8 Network topology: **(a)** EON with 28 nodes and 68 directed links; **(b)** AABN with 27 nodes and 74 directed links

request holding coefficient at nodes, the average throughput becomes larger. Moreover, whether simulated on the AABN or EON, our proposed QoS Aware Routing (QAR) algorithm performs more efficiently than the other three algorithms. In addition, in the simulation of these two networks, the average throughput on the EON is almost twice as large as it on the AABN. This is mainly because that the total simulation time on the EON is less than on the AABN no matter which routing algorithm is used.

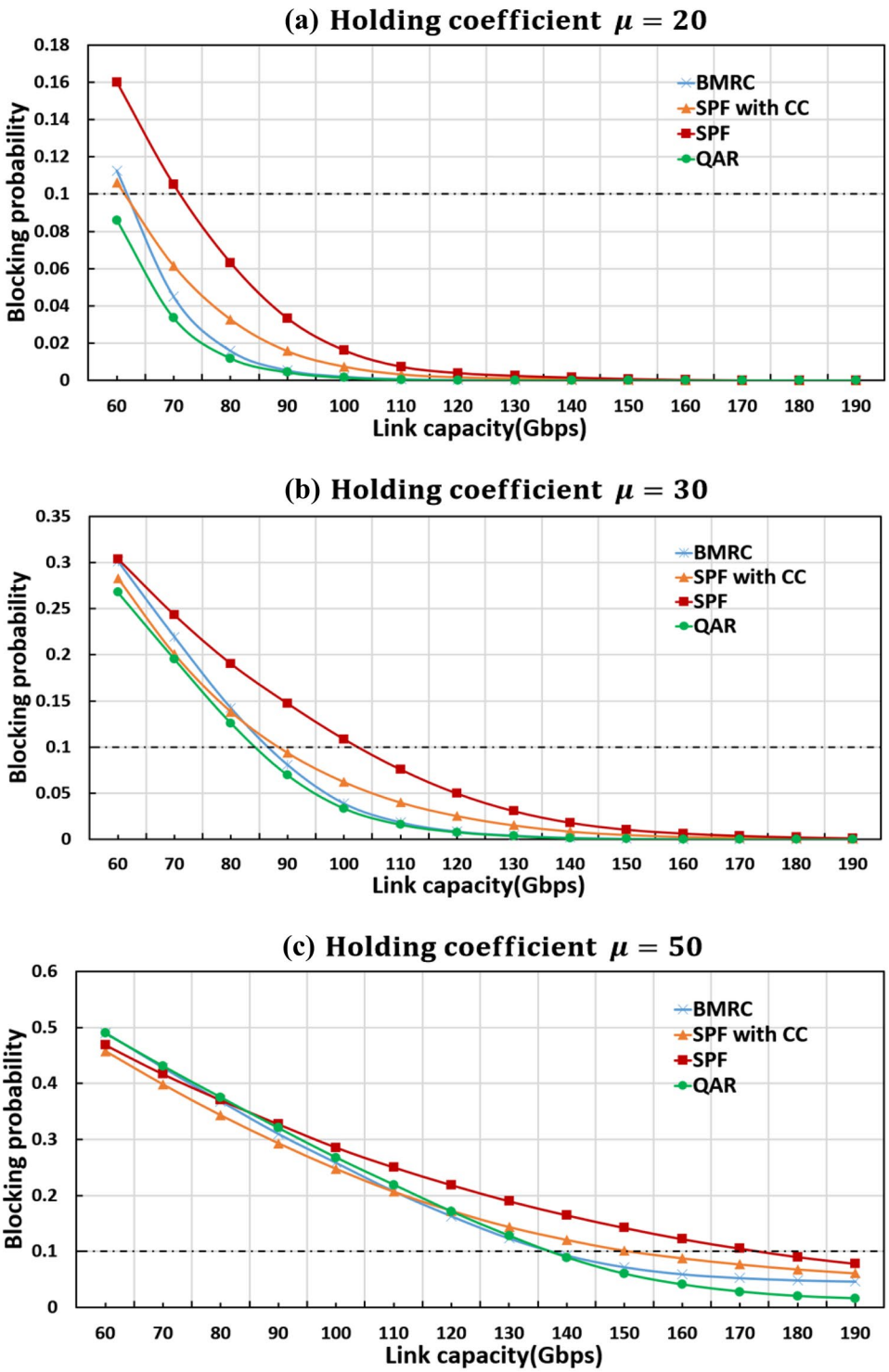


Fig. 9 Blocking probability on American ATT Backbone Network (AABN) with various holding coefficients

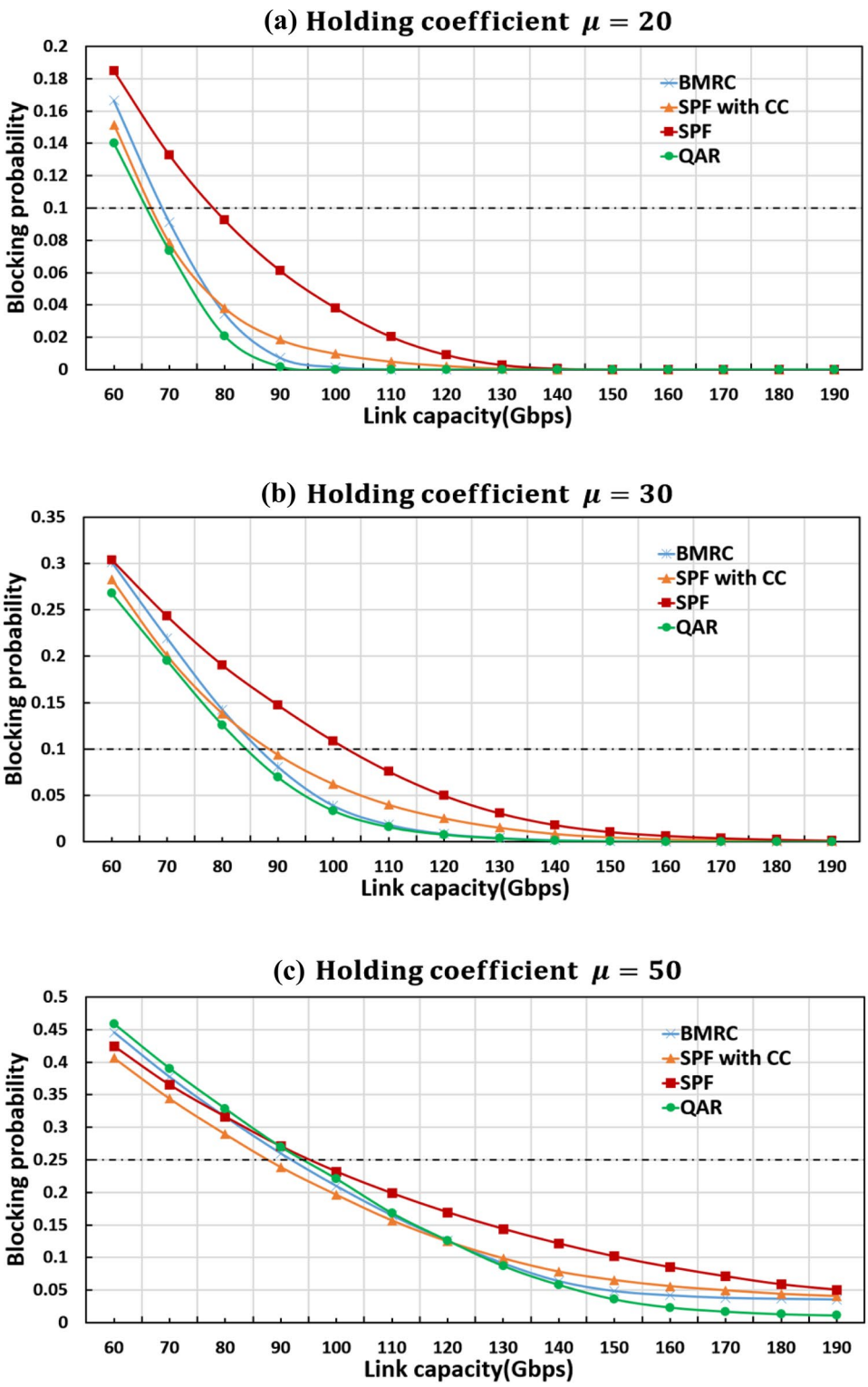


Fig. 10 Blocking probability on the European Optical Network (EON) with various holding coefficients

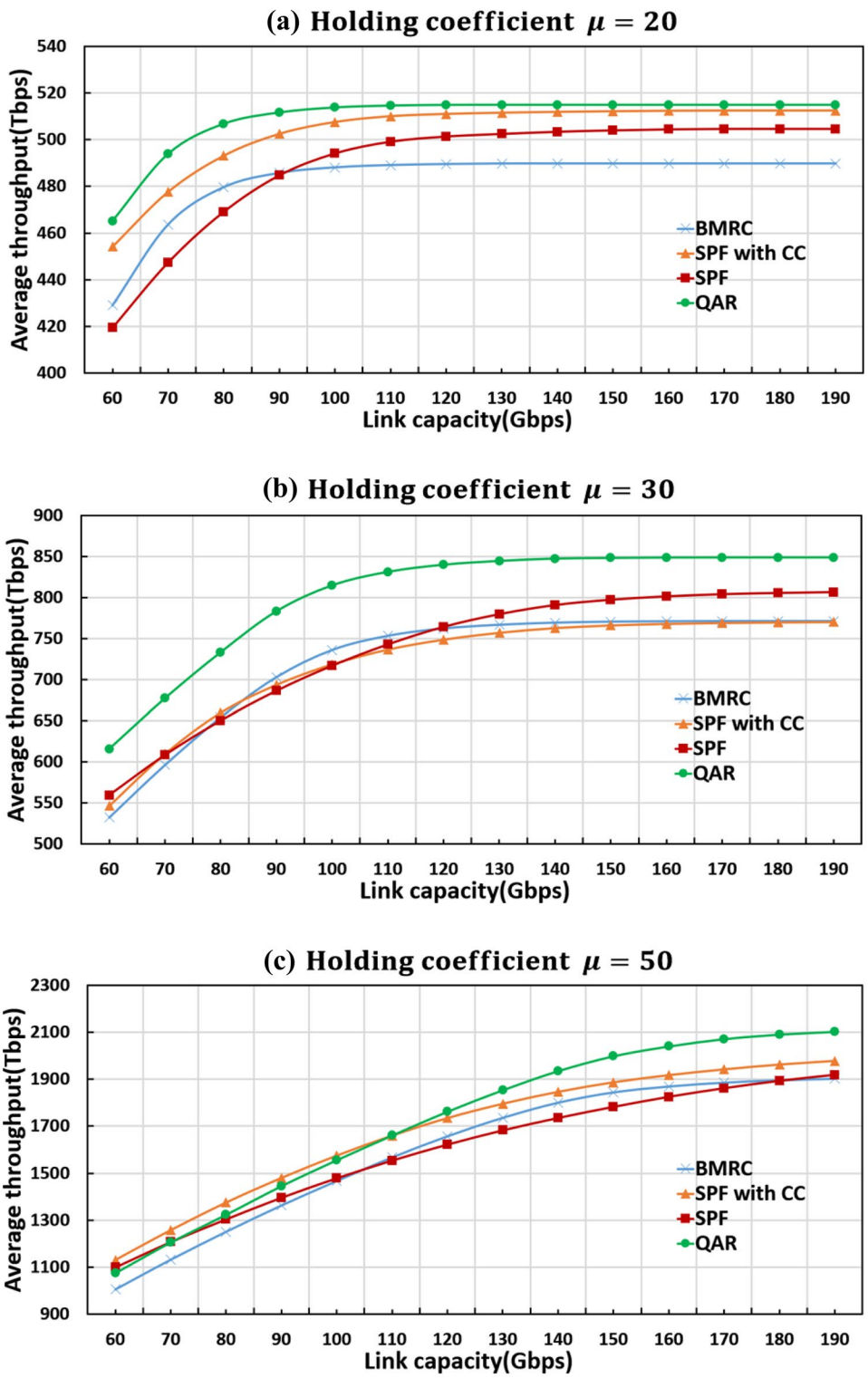


Fig. 11 Average throughput on the American ATT Backbone Network (AABN) with various holding coefficients

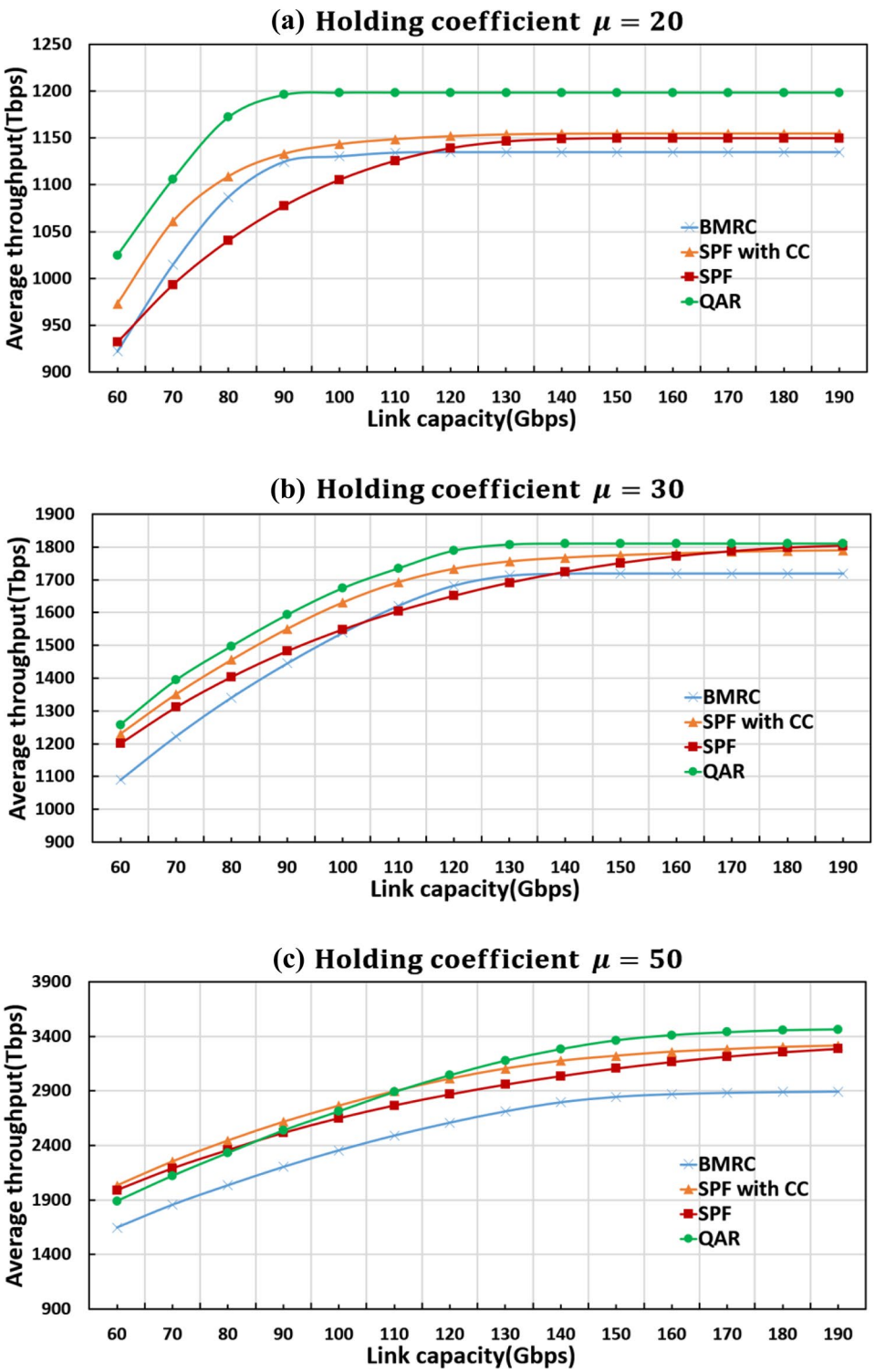


Fig. 12 Average throughput on European Optical Network (EON) with various holding coefficients

5 Conclusion

The main contribution of this paper is to apply the Deep Neural Networking (DNN) as a traffic QoS requirements classification technique to SDN routing. We proposed an ILP-selected feature reduction method to preprocess the data and test it by comparing the classification accuracy with previous works. The classification accuracy of DNN with different feature reduction methods shows that our proposed approach obtains better results than previous studies. In order to

improve routing performance, we also put forward a heuristic algorithm, called QoS aware routing (QAR) algorithm. This algorithm considers the QoS requirements of each request (both transmission latency and bandwidth requirements) which have been classified by the DNN model. The simulation results indicate that our proposed QAR algorithm performs better than other previous routing algorithms. Overall, our findings indicate that reflecting QoS requirements in routing decisions can enable remarkable routing efficiency and cost savings. To further improve routing performance, future work should consider other QoS requirement parameters.

Acknowledgements We would like to thank Prof. Neil Millar and anonymous reviewers, whose insightful comments and modification suggestions are valuable for the improvement and completion of this paper. This work has been partially supported by JSPS Grant-in-Aid for Scientific Research (C) 21K04544.

References

1. Optimization G (2016) Gurobi optimizer[J]. https://www.gurobi.com
2. Rojas JS (2017) Ip network traffic flows labeled with 75 apps. *Kaggle*. Retrieved from https://www.kaggle.com/jsrojas/ip-network-traffic-flows-labeled-with-87-apps
3. Akin E, Korkmaz T (2019) Comparison of routing algorithms with static and dynamic link cost in SDN. In 16th IEEE Annual Consumer Communications & Networking Conference (CCNC) pp. 1–8
4. Alam F, Katib I, Alzahrani AS (2013) New networking era: Software defined networking. Int J Adv Res Comput Sci Softw 3(11):349–353
5. AlGhadhban A, Shihada B (2018) Flight: A fast and lightweight elephant-flow detection mechanism. In 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS) pp. 1537–1538
6. Amaral P, Dinis J, Pinto P, Bernardo L, Tavares J, Mamede HS (2016) Machine learning in software defined networks: data collection and traffic classification. In Proc 24th Int Conf Network Protocols (ICNP) pp. 1–5
7. Astuto B, Mendonca M, Nguyen Z, Obraczka K, Turletti T (2014) A survey of software-defined networking: Past, present, and future of programmable networks. IEEE Commun Surv Tutorials 16(3):1617–1634
8. Becker N, Werft W, Toedt G, Lichter P, Benner A (2009) Penalizedsvm: A r-package for feature selection svm classification. Bioinformatics 25(13):1711–1712
9. Biau G, Scornet E (2016) A random forest guided tour. Test 25(2):197–227
10. Callado AC, Kamienski CA, Szabó G, Gero BP, Kelner J, Fernandes SF, Sadok DFH (2009) A survey on Internet traffic identification. IEEE Commun Surv Tutorials 11(3):37–52
11. Caruana R, Niculescu-Mizil A (2006) An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd International Conference on Machine learning ACM pp. 161–168
12. Chen Y, Farley T, Ye N (2004) QoS requirements of network applications on the internet. Inf Knowl Syst Manag 4(1):55–76
13. Chhabra A, Kiran M (2017) Classifying elephant and mice flows in high-speed scientific networks
14. Christiansen B (2005) The shortcomings of nonlinear principal component analysis in identifying circulation regimes. J Clim 18:4814–4823
15. Conti M, Gregori E, Panzieri F (2000) Load distribution among replicated web servers: A qos-based approach. ACM SIGMETRICS Performance Evaluation Review 27(4):12–19
16. Da Silva AS, Machado CC, Bisol RV, Granville LZ, Schaeffer-Filho A (2015) Identification and selection of flow features for accurate traffic classification in SDN. In Proc IEEE 14th Int Symp Network Computing and Applications (NCA) IEEE pp. 134–141
17. Erman J, Arlitt M, Mahanti A (2006) Traffic classification using clustering algorithms. In Proceedings of the 2006 SIGCOMM workshop on Mining network data pp. 281–286
18. Erman J, Mahanti A, Arlitt M, Cohen I, Williamson C (2007) Semi-supervised network traffic classification. In Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer ystems pp. 369–370
19. Ersoz D, Yousif MS, Das CR (2007) Characterizing network traffic in a cluster-based, multi-tier data center. In 27th International Conference on Distributed Computing Systems (ICDCS'07) IEEE pp. 59
20. Fonti V, Belitser E (2017) Feature selection using lasso. VU Amsterdam Research Paper in Business Analytics 30:1–25
21. Jarschel M, Zinner T, Höhn T, Tran-Gia P (2013) On the accuracy of leveraging sdn for passive network measurements. In 2013 Australasian Telecommunication Networks and Applications Conference (ATNAC) IEEE pp. 41–46
22. Jolliffe I (2002) Principal Component Analysis, 2nd ed. Springer
23. Karakus M, Durresi A (2017) Quality of service (QoS) in software defined networking (SDN): A survey. J Netw Comput Appl 80:200–218
24. Kotsiantis SB, Zaharakis I, Pintelas P (2007) Supervised machine learning: A review of classification techniques. Emerg Artif Intell Appl Comput Eng 160:3–24
25. Layeghy S, Pakzad F, Portmann M (2016) Scor: Software-defined constrained optimal routing platform for sdn. arXiv preprint arXiv:1607.03243
26. Li C-Y, Li G, Wai P-KA, Li VO (2004) A wavelength-switched time-slot routing scheme for wavelength-routed networks. In 2004 IEEE International Conference on Communications (IEEE Cat. No. 04CH37577) 3:1689–1693
27. Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE (2017) A survey of deep neural network architectures and their applications. Neurocomputing 234:11–26
28. McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J (2008) Openflow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review 38(2):69–74
29. Menze BH, Kelm BM, Masuch R, Himmelreich U, Bachert P, Petrich W, Hamprecht FA (2009) A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data. BMC Bioinformatics 10(1):1–16
30. Mwangi B, Tian TS, Soares JC (2014) A review of feature reduction techniques in neuroimaging. Neuroinformatics 12(2):229–244
31. Neums L, Meier R, Koestler DC, Thompson JA (2019) Improving survival prediction using a novel feature selection and feature reduction framework based on the integration of clinical and molecular data. In PACIFIC SYMPOSIUM ON BIOCOMPUTING 2020. World Scientific, pp. 415–426
32. Patle A, Chouhan DS (2013) Svm kernel functions for classification. In 2013 International Conference on Advances in Technology and Engineering (ICATE) IEEE pp. 1–9
33. Pua Y-H, Kang H, Thumboo J, Clark RA, Chew ES-X, Poon CL-L, Chong H-C, Yeo S-J (2019) Machine learning methods are comparable to logistic regression techniques in predicting severe walking limitation following total knee arthroplasty. Knee Surgery, Sports Traumatology, Arthroscopy pp. 1–10
34. Ripley BD (2007) Pattern recognition and neural networks. Cambridge University Press
35. Rish I et al (2001) An empirical study of the naive bayes classifier. In IJCAI 2001 workshop on empirical methods in artificial intelligence 3:41–46

36. Shlens J (2014) A tutorial on principal component analysis. arXiv preprint arXiv:1404.1100
37. Sylvester EV, Bentzen P, Bradbury IR, Clément M, Pearce J, Horne J, Beiko RG (2018) Applications of random forest feature selection for fine-scale genetic population assignment. Evol Appl 11(2):153–165
38. Tibshirani RJ et al (2013) The lasso problem and uniqueness. Electron J Stat 7:1456–1490
39. Velasco L, Jirattigalachote A, Ruiz M, Monti P, Wosinska L, Junyent G (2012) Statistical approach for fast impairment-aware provisioning in dynamic all-optical networks. J Opt Commun Networking 4(2):130–141
40. Vu HD, But J (2015) How rtt between the control and data plane on a sdn network impacts on the perceived performance. In 2015 International Telecommunication Networks and Applications Conference (ITNAC) IEEE pp. 179–184
41. Wang Z, Crowcroft J (1996) Quality-of-service routing for supporting multimedia applications. IEEE J Sel Areas Commun 14(7):1228–1234
42. Wei JY, McFarland RI (2000) Just-in-time signaling for wdm optical burst switching networks. J Lightwave Technol 18(12):2019–2037
43. Xiao P, Liu N, Li Y, Lu Y, Tang X-J, Wang H-W, Li M-X (2016) A traffic classification method with spectral clustering in SDN. In Proc 17th Int Conf Parallel and Distributed Computing, Applications and Technologies (PDCAT) IEEE pp. 391–394
44. Xie J, Yu FR, Huang T, Xie R, Liu J, Liu Y (2018) A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges. IEEE Commun Surv Tutorials 21(1):393–430
45. Yamansavascilar B, Guvensan MA, Yavuz AG, Karsligil ME (2017) Application identification via network traffic classification. In 2017 International Conference on Computing, Networking and Communications (ICNC) IEEE pp. 843–848
46. Yang M, Wu Q, Guo K, Zhang Y (2019) Evaluation of device cost, power consumption, and network performance in spatially and spectrally flexible sdm optical networks. J Lightwave Technol 37(20):5259–5272
47. Ye Q, Li J, Qu K, Zhuang W, Shen X, Li X (2018) A network slicing framework for end-to-end QoS provisioning in 5G networks. IEEE Veh Technol Mag 13(2):65–74
48. Yen JY (1971) Finding the k shortest loopless paths in a network. Manag Sci 17(11):712–716
49. Zhang J, Chen X, Xiang Y, Zhou W, Wu J (2014) Robust network traffic classification. IEEE/ACM Trans Networking 23(4):1257–1270
50. Zhang J, Xiang Y, Wang Y, Zhou W, Xiang Y, Guan Y (2012) Network traffic classification using correlation information. IEEE Trans Parallel Distrib Syst 24(1):104–117

Weichang Zheng Weichang Zheng is studying for Ph.D. degree at the Graduate School of Systems and Information Engineering.

Mingcong Yang Mingcong Yang received the Ph.D. degree from the Graduate School of Systems and Information Engineering.

Chenxiao Zhang Chenxiao Zhang is studying for Ph.D. degree at the Graduate School of Systems and Information Engineering.

Yu Zheng Yu Zheng is studying for M.S. degree at the Graduate School of Systems and Information Engineering.

Yunyi Wu Yunyi Wu is studying for Ph.D. degree at the Graduate School of Systems and Information Engineering.

Yongbing Zhang Yongbing Zhang is a Professor at the Graduate School of Systems and Information Engineering. His research include distributed computer systems, communication networks, and performance evaluation.

Jie Li Jie Li is a professor at Department of Computer Science and Engineering. His research include big data, cloud and Edge computing, machine learning and networking.