A Machine Learning Approach for Traffic Flow Provisioning in Software Defined Networks

Subham Kumar
Department of Computer Science
BITS Pilani
Pilani, India
h20180123@pilani.bits-pilani.ac.in

Gaurang Bansal
Department of Computer Science
BITS Pilani)
Pilani, India
h20180123@pilani.bits-pilani.ac.in

Virendra Singh Shekhawat
Department of Computer Science
BITS Pilani
Pilani, India
vsshekhawat@pilani.bits-pilani.ac.in

***Abstract*—With the recent surge of machine learning and artificial intelligence, many research groups are applying these techniques to control, manage, and operate networks. Software Defined Networks (SDN) transform the distributed and hardware-centric legacy network into an integrated and dynamic network that provides a comprehensive solution for managing the network efficiently and effectively. The network-wide knowledge provided by SDN can be leveraged for efficient traffic routing in the network. In this work, we explore and illustrate the applicability of machine learning algorithms for selecting the least congested route for routing traffic in a SDN enabled network. The proposed method of route selection provides a list of possible routes based on the network statistics provided by the SDN controller dynamically. The proposed method is implemented and tested in Mininet using Ryu controller.**

***Index Terms*—Traffic Engineering, Machine Learning, Software Defined Networking, Clustering**

I. Introduction

Traditional networks are inherently distributed in nature, and they rely on hop-based routing techniques to route and forward packets. The least hop path between any two nodes is chosen, and then all traffic is routed along that path. This technique of sending traffic worked fine when networks were designed initially, but as they grew in size and usage, problems like congestion started popping up, choking the links and rendering communication ineffective. Routing protocols like hop-based methods do not incorporate the present level of congestion in the network into their route calculations due to large control overhead incurred for sharing updated network states in distributed manner. However, this often leads to inefficient utilization of network resources, as packets can be routed through a slightly longer path which is less congested compared to the congested least hops path, leading to better traffic load distribution in the network. Software Defined Networking (SDN) is a new networking paradigm which enables centralized control over the network infrastructure by separating control and data planes [1]. It provides a programmable interface between network devices and the centralized controller. The centralized controller provides a global view of the entire network that gives more flexibility for controlling and operating the network to meet the desired Quality of Service (QoS) requirement efficiently. Such logical global view of the network is missing in traditional networks.

Machine Learning is being used extensively in a variety of applications and fields today. Supervised machine learning algorithms are used to predict future events using labeled examples. By analyzing a known training dataset, the algorithm infers an appropriate function to provide relevant results. In contrast, unsupervised machine learning algorithms are used when there is no prior information available to train the algorithm [2]. It tries to draw inferences from the dataset in order to describe hidden similarities and characteristics present in the unlabeled data. Semi-supervised machine learning algorithms use a combination of both labeled and unlabeled data for training. This ensemble can be used for considerable improvement in learning accuracy. Reinforcement machine learning algorithms generate actions and modify them corresponding to penalties or rewards. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance.

Machine learning techniques are hard to apply and deploy in traditional networks to control and operate networks due to their distributed and dumb nature. The SDN brings us new opportunities to incorporate intelligence inside the networks [3]. The capabilities of SDN (for example centralized control, global view of the network, software-based traffic control amp; analysis, and dynamic traffic routing) make it easier to apply machine learning techniques. This paper proposes a machine learning-based congestion aware traffic routing method in the networks.

II. Previous work and overview

As networks are growing, operation and management of such complex and variegated networks is becoming increasingly complicated for network administrators. There is a lot of research effort being put to find solutions that can help in better management and agility of networks without impacting its performance. Various approaches have been attempted to address this issue, like distributed algorithms in self healing [4]. Contemporary research can be categorized primarily into two categories for a better overview :

1) Feature Traffic Forecasting (FTF)
2) Time Traffic Forecasting (TTF)

Feature Traffic Forecasting (FTF) is based on understanding the network characteristics and extracting certain parameters of the network such as total flow present in the network and how users are correlated to each other. FTF deploys supervised and unsupervised learning with relatively high accuracies, deploying Artificial Neural Networks [5] and in Intrusion Detection Systems [6].

Time Traffic Forecasting utilizes the network topology and the amount data flowing in the network as shown in [7] and [8].Time Traffic Forecasting (TTF) aims to construct a model that can draw accurate correlation between future traffic volume and previously observed traffic volumes. TTF models can be categorised into:

1) Statistical Analysis model (SAM)
2) Network Analytics Supervised Learning (NASL)

NASL is designed to process traffic patterns based on what it has learnt from the dataset it was trained on. However this has a limitation, which is explained as follows. Each network is different, the flows and the amount of data that is flowing is different. So what works well for one network may be result in performance degradation for another. So researchers are working on SAM where the model learns the network characteristics while being deployed in real time without adding to network latency.

The major drawback of machine learning is that these techniques cannot be deployed in real time easily. On one hand machine learning provides high accuracies on analysis of network traffic, they have been deployed extensively in Intrusion detection systems as shown by [9]. There have been some applications in circuit switched networks as well[10]. However this is at expense of computational effort and time, as illustrated by [11] . The goal is to design a real time system which may come at the expense of a slight loss in accuracy. For instance [12] have tried logistic regression, but this does not have any mechanism to effectively incorporate new states, and the running time becomes significant when the network size increases.

An alternative to conventional machine learning approaches is reinforcement learning. Reinforcement learning has been applied to networking for a long time, [13] have applied intelligent learning modules inside routers, to try and incorporate dynamic parameters like congestion into deciding the flow of packets, however they could achieve limited success because the experimental runs involved are too costly, and the algorithm employs a greedy approach, resulting in the selection of suboptimal paths more frequently as congestion increases, especially in larger networks. Deep reinforcement learning can help mitigate these issues to a certain extent [14]. However experiments conducted by [15] shows how the computationally intensive nature of these algorithms proves to be a roadblock in effective deployment of such techniques in the domain of networking.

In Software defined Networking (SDN), a centralized controller decides the flow of data through the network. A centralized controller enables the availability of information about the whole network at one place. This centralized controller collects various statistics as soon as the state of the network changes. Each of the switches sends a control packet to the controller as soon as it detects a change in the link attributes. This information is aggregated over all the link values and assimilated by the controller. Available bandwidth (dynamic, at a particular instant) is one of the most important metrics for effective dynamic traffic engineering, especially when we consider factors like congestion [16].

III. Proposed Methodology

This paper proposes a machine learning based path selection approach for traffic flow provisioning in software defined networks. The proposed approach comprises of two modules, training and deployment. The training module learns from the paths provisioned in the recent past for the given state of the network. In the deployment module, the controller queries the module at specified interval of time for the best possible path based on the current network state and then provisions the new paths based on the information received. The machine learning module adapts to the network topology and makes intelligent decisions for the traffic flows routing. The traffic ows are routed in the network considering the congestion and traffic pattern history in the network. The state of the network is assimilated at the SDN controller. The controller generates a bandwidth matrix for each of the links after calculating the available bandwidth. The available bandwidth at each of the links in the network is provided as input to the deployment module.

We have proposed two methods for selecting best possible path to route the traffic flow, in the network. One method is based on K-means clustering and other uses a novel application of the Vector Space Model with cosine similarity.

K-Means clusters a given dataset into K clusters, based on euclidean distance. In this work it is used for clustering similar states of the network based on euclidean distance between them. The network state is represented through weights which are assigned to each of the links. In the training module, every state of the network along with its corresponding best path is read, and appropriately put into existing clusters, or the clusters are rearranged to form a new set of clusters. The best paths corresponding to each state are provided by the network administrator, during the training module. At the end of the training module, we will have a set of K clusters, with each cluster corresponding to similar network states, that is network states which give similar best paths.

Cosine similarity has been used extensively in the domains of Information Retrieval and Data Mining[17]. It finds the similarity between two points based on the angle between the two vectors which connect the origin to those points respectively. The network states are clustered in a similar fashion as in K-Means, but using cosine similarity as the clustering parameter instead of Euclidean distance. Network states which are similar will be aligned with each other to a good degree, resulting in a high value of cosine similarity compared to a state which is quite different (considerable difference in the best paths).

The hop-based routing algorithm (e.g., Dijkstra's algorithm) used in conventional networks (non-SDN) calculates the least

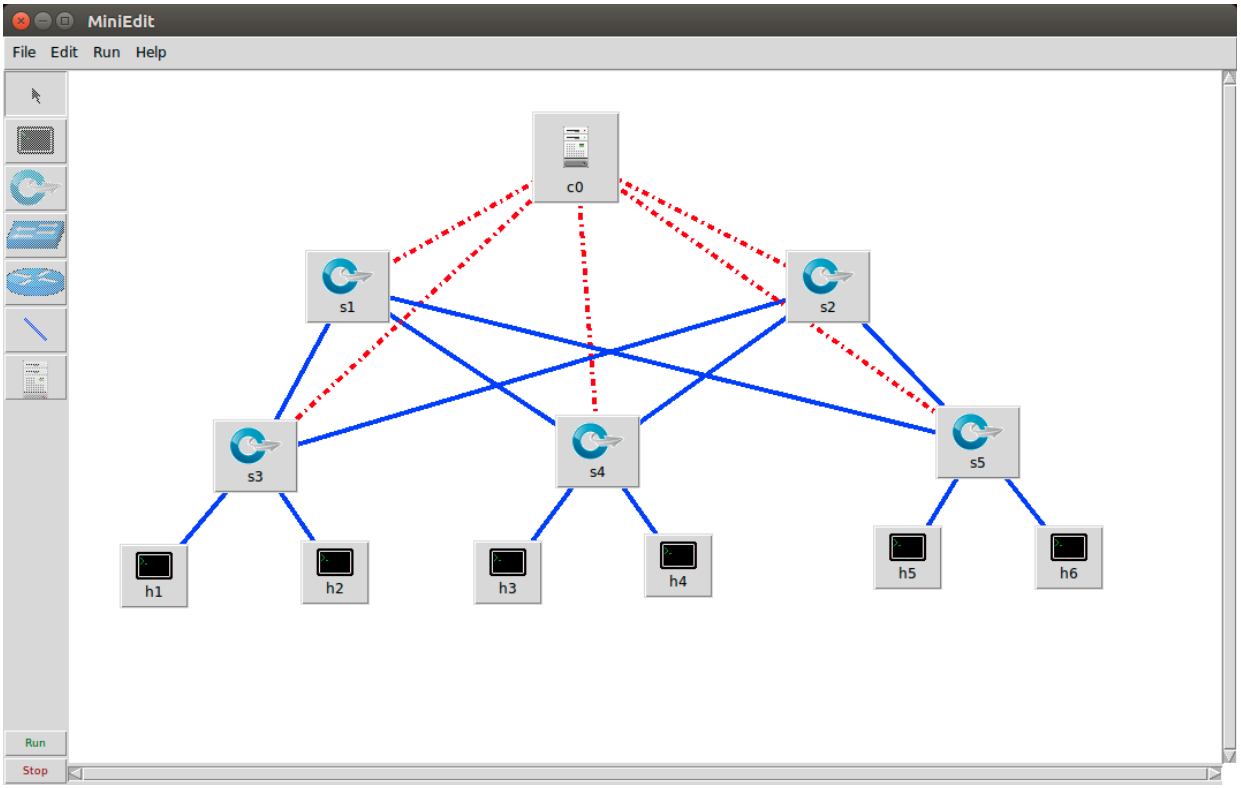

Fig. 1: Test network topology

cost path between source and destination nodes, where cost for each of the links in the network is defined as one. The hop-based routing does not account for any dynamic parameters like link congestion for traffic routing. In conventional networks, traffic routing based on such dynamic network parameters creates a huge amount of control traffic. As a result the hop-based routing method cannot avoid congested links from the path selected for a traffic flow, which leads to poor performance.

The main advantage of K-Means and cosine similarity is the finite and constant number of comparisons which have to be made to find the best path. Even though there can be a very large number of network states, because they have been clustered based on the best paths, the search space for comparisons is dramatically reduced, and this enables us to use dynamic parameters like congestion occurring in the network as the link weights, which can change frequently. A change in just one of the links will cause the Djikstra algorithm to be run for the entire network again, but in when these machine learning models are used, a few comparisons have to be made, which is much more faster.

Unsupervised techniques have an advantage in this application that they can incorporate information which they have not seen before as well into their learning. For example if there is a network state which is very different from the network states whose information has been incorporated into the model in the training module, this state and the corresponding best path can be added and the clusters adjusted appropriately, which would not have been possible in conventional supervised approaches.

Table I provides the complexity of different Machine Learning algorithms, where m is the number of training examples, which is analogous to the number of network states utilized in the training module for our work and n is the size of input. The parameter c refers to the number of output classes, and for K-Means, it represents the number of clusters.

Even though some algorithms like neural networks can give better results, the computational costs and running time are too high for the controller to be able to reroute the packets via an alternate path. The difference becomes starker when the network topology increases in size. Since time is one of the most important factors which has to be minimized, we used K-Means and similar clustering algorithms which have the least time complexity for classification of network states.

TABLE I: Complexity of Different Machine Learning Algorithms

Algorithm	Complexity
Logistic Regression	O(mn$^2 + n^3$)
PCA	O(mn$^2 + n^3$)
SVM	O(m$^2 n$)
EM	O(mn$^2 + n^3$)
K-Means	O(mnc)
Neural Networks	O(mn + nc)

Cosine similarity performs better than K-Means in certain cases where cosine similarity proves to be a better metric than euclidean distance. An example is illustrated in Figure

2. (a,b,c) refers to the weights of link a,b and c respectively as shown.

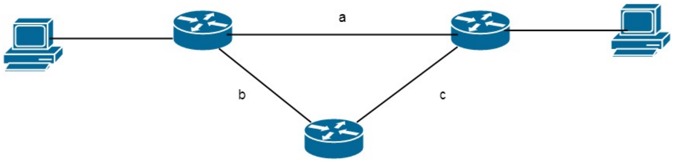

Fig. 2: Simple network layout

Let's say that we have 2 clusters in the training module, with centres (4,4,4) and (3,1,1) respectively. The best path between the two host machines in the first cluster is a, while for the second cluster is bc. A test path, (2,2,2) has to be mapped to either of these them in the deployment module. (2,2,2) should be mapped to (4,4,4) because it is similar in nature to that state, giving similar best paths. However Euclidean Distance will return (3,1,1) as the closest, rather than (4,4,4). But on the other hand cosine similarity will give the highest value for (4,4,4) rather than (3,1,1) and will return (4,4,4) as the result, this is illustrated in Figure 3. Hence cosine similarity will perform better in such scenarios compared to K-Means.

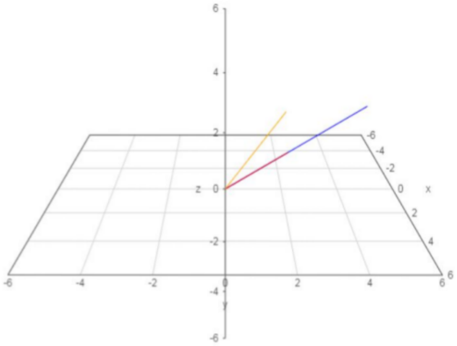

Fig. 3: Cosine Similarity vs K-Means

IV. Experiments and Results

The test network topology is shown in Figure 1. This was created in MiniEdit, which is a GUI running on top of MiniNet. The Ryu controller has been utilised for this experiment.Each of the links are given weights which are calculated as the ratio of Available Bandwidth to Total Bandwidth on that particular link. We analyze communication taking place between host machines H1 and H4. Network traffic will be simulated by generating communication traffic using other hosts. Traffic is generated between the other hosts in a varying fashion, such that the links on all possible paths between H1 and H4 have varying weights at different instances of time.

The statistics (state) of the network is assimilated at the SDN controller every 0.1 seconds.This time frame can also be adjusted as per the state of the network, if there are signs of congestion then this duration can be shortened to adapt effectively to the state of the network. At other times it can be kept at a suitable steady interval, this varies according to the network topology and size. After calculating the available bandwidth, the controller generates a bandwidth matrix. This matrix contains information about each of the links and the available bandwidths in each of them. This is given as input to the machine learning module. The module returns the best path to the controller, which translates and pushes the information down into the switches.

In the first run we clustered the paths using K-Means into 5 clusters, based on the common links between them. Various experiments were carried out with different values of K to find the best value. Then the input was compared to the respective centroids of each of the clusters, which in this case are similar paths. Then the comparison is done with the paths present within that cluster, again giving us a list of possible paths, rankwise. This approach is much more efficient compared to an approach where a comparison is made between the test path and all possible paths . In this case, we will be comparing a small subset of the search space, just 5% (1 cluster out of 5) of all paths which have been stored.

In the second run, we construct a n-dimensional space where each dimension represents a link in the network topology, and this way all the possible paths are marked. Then the input is also plotted and compared to them. The difference is that here we are computing cosine similarity, which is a better parameter as compared to Euclidean distance which does not always give a proper estimate, especially if the input is a multiplied factor of one of the paths.

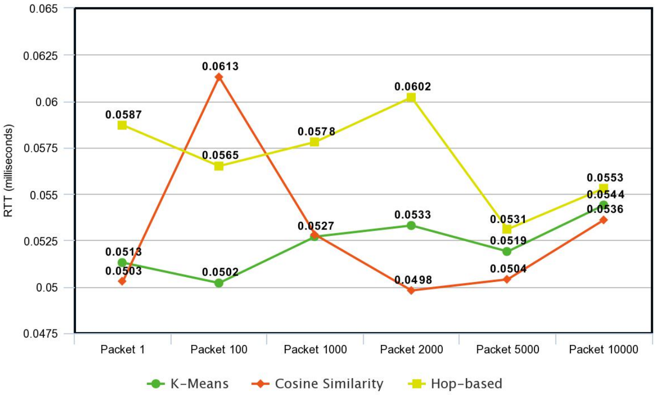

Fig. 4: Comparison of RTT for various packets taken by K-Means, Cosine Similarity and the hop-based routing algorithm

The Round Trip Time (RTT) measurement of the traffic flows for three different path selection techniques, i.e., K-means, Cosine Similarity, and Hop based routing is shown in Figure 4. The RTT measurements have been taken for the transmission of a specific packet number at different time intervals. It is evident from the plot that overall the paths selected by the K-means and Cosine similarity techniques provides lower RTT paths as compared to conventional Hop based method.

However, initially (number of packets transmitted are around 100) Cosine similarity gives higher RTT paths as compared to other two path selection methods. But as time progresses and more number of packets get transmitted, Cosine similarity method selects better paths in terms of RTT as compared to K-means as well as hop based routing method. With the increase in number of packets new paths are added into the repository due to quick changes into the network congestion state. The Cosine similarity and K-means methods are taking into account the path repository and selects a path based on the current network congestion state. Hop based method does not incorporate the network congestion state changes into the path selection process. Therefore, the paths selected by hop based method are inferior in terms of RTT when network is congested.

Further increase in the network congestion due to more traffic leads to increase in the RTT values for all of the three paths selection methods, which can be visualized in the Figure 4 for RTT measurements taken for 10000th packet. This increase observed in RTT for all path selection methods is due to the network resource limitations, which means network resources are insufficient to handle the amount of traffic generated in the network. Cosine Similarity provides lower RTT values compared to K- Means, because of reasons explained using an example topology shown in Figure 2.

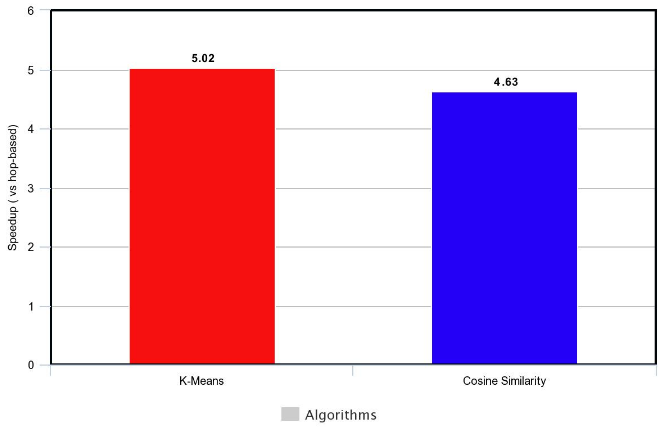

Fig. 5: Speedup of various stated algorithms compared to conventional routing methods

The speedup comparison between K-means and Cosine similarity with hop based has been shown in the Figure 5. The results confirms the superiority of K-means over Cosine similarity with better speedup ratio.

The graph plotted in Figure 6 shows the performance comparison of K-Means and Cosine Similarity to the best possible (optimal or ground truth) solution in terms of the average RTT, minimum RTT, maximum RTT and the mean deviation of the RTT. When we consider network as a whole for a considerable amount of time, then it can be seen that both methods select best possible paths or very close to that. However, Cosine similarity gives us a higher accuracy at the cost of a little less speedup as compared to K-Means.

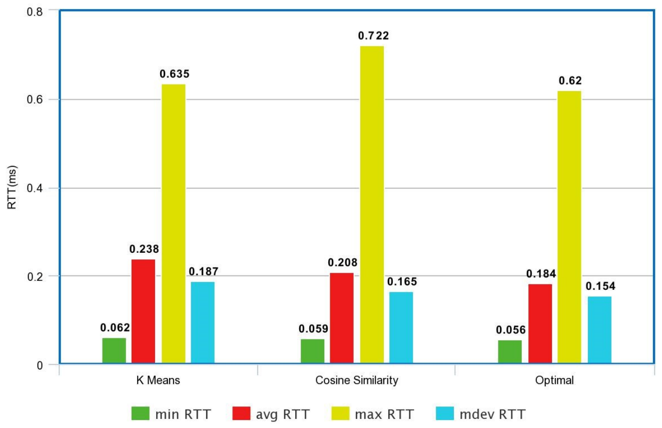

Fig. 6: Comparison of K-Means and Cosine Similarity to the ground truth (optimal solution)

V. Conclusions and Future work

In this paper, we have proposed a machine learning model for congestion aware traffic routing in software defined networks. The proposed model learns updated network statistics at different intervals of time and incorporates that knowledge for selecting least congested path for traffic flow routing. The least congested path is selected from a list of probable paths by using K-Means and Cosine Similarity methods. The experimental results show that paths selected by using Cosine Similarity are better in terms of RTT value as compared to K-Means. However, K-Means outperforms Cosine Similarity for the speedup ratio, which is calculated with respect to conventional hop-based routing method. In future, this model can be extended to apply for other traffic management functionalities like load balancing, traffic load distribution and multi-path routing.

References

[1] D. Kreutz, F. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *arXiv preprint arXiv:1406.0440*, 2014.

[2] N. Grira, M. Crucianu, and N. Boujemaa, "Unsupervised and semi-supervised clustering: a brief survey," *A review of machine learning techniques for processing multimedia content*, vol. 1, pp. 9–16, 2004.

[3] T. Abar, A. B. Letaifa, and S. El Asmi, "Machine learning based qoe prediction in sdn networks," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2017, pp. 1395–1400.

[4] A. Trehan, "Algorithms for self-healing networks," *CoRR*, vol. abs/1305.4675, 2013. [Online]. Available: http://arxiv.org/abs/1305.4675

[5] Y. Li, H. Liu, W. Yang, D. Hu, and W. Xu, "Inter-data-center network traffic prediction with elephant flows," in *2016 IEEE/IFIP Network Operations and Management*

Symposium, NOMS 2016, Istanbul, Turkey, April 25-29, 2016, 2016, pp. 206–213. [Online]. Available: https://doi.org/10.1109/NOMS.2016.7502814

[6] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016. [Online]. Available: https://doi.org/10.1109/COMST.2015.2494502

[7] H. Derbel, N. Agoulmine, and M. Salaün, "ANEMA: autonomic network management architecture to support self-configuration and self-optimization in IP networks," *Computer Networks*, vol. 53, no. 3, pp. 418–430, 2009. [Online]. Available: https://doi.org/10.1016/j.comnet.2008.10.022

[8] A. J. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis," *Commun. ACM*, vol. 55, no. 2, pp. 55–61, 2012. [Online]. Available: https://doi.org/10.1145/2076450.2076466

[9] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE symposium on security and privacy*. IEEE, 2010, pp. 305–316.

[10] L. Li, Y. Zhang, W. Chen, S. K. Bose, M. Zukerman, and G. Shen, "Naïve bayes classifier-assisted least loaded routing for circuit-switched networks," *IEEE Access*, vol. 7, pp. 11 854–11 867, 2019.

[11] Y. Zuo, Y. Wu, G. Min, and L. Cui, "Learning-based network path planning for traffic engineering," *Future Generation Computer Systems*, vol. 92, pp. 59–67, 2019.

[12] S. Troia, A. Rodriguez, I. Martín, J. A. Hernández, O. G. De Dios, R. Alvizu, F. Musumeci, and G. Maier, "Machine-learning-assisted routing in sdn-based optical networks," in *2018 European Conference on Optical Communication (ECOC)*. IEEE, 2018, pp. 1–3.

[13] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Advances in neural information processing systems*, 1994, pp. 671–678.

[14] G. Stampa, M. Arias, D. Sanchez-Charles, V. Muntés-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," *arXiv preprint arXiv:1709.07080*, 2017.

[15] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 393–430, 2018.

[16] R. Jain, "Congestion control in computer networks: Trends and issues," *CoRR*, vol. cs.NI/9809091, 1998. [Online]. Available: http://arxiv.org/abs/cs.NI/9809091

[17] T. Korenius, J. Laurikkala, and M. Juhola, "On principal component analysis, cosine and euclidean measures in information retrieval," *Information Sciences*, vol. 177, no. 22, pp. 4893–4905, 2007.