MER-SDN: Machine Learning Framework for Traffic Aware Energy Efficient Routing in SDN

Beakal Gizachew Assefa and Oznur Ozkasap
Department of Computer Engineering
Koc University, Istanbul, Turkey
{bassefa13, oozkasap}@ku.edu.tr

Abstract—**Software Defined Networking (SDN) achieves programmability of a network through separation of the control and data planes. It enables flexibility in network management and control. Energy efficiency is one of the challenging global problems which has both economic and environmental impact. A massive amount of information is generated in the controller of an SDN based networks. Machine learning gives the ability to computers to progressively learn from data without having to write specific instructions. In this work, we propose MER-SDN: a machine learning framework for traffic aware energy efficient routing in SDN. Feature extraction, training, and testing are the three main stages of the learning machine. Experiments are conducted on Mininet and POX controller using real-world network topology and dynamic traffic traces from SNDlib. Results show that our approach achieves more than 65% feature size reduction, more than 70% accuracy in parameter prediction of an energy efficient heuristics algorithm, also our prediction refine heuristics converges the predicted value to the optimal parameters values with up to 25X speedup as compared to the brute force method.**

I. INTRODUCTION

SDN is a widely accepted networking paradigm based on the concept of separation of control and data planes. It is implemented in a home network, campus networks, ISP, telecom, and cloud data centers. Major companies like Facebook, Yahoo, Microsoft, Huawei, Cisco, and Google has adopted SDN to their data centers and network equipment designs [1], [2].

Machine learning is used in various disciplines as a tool to discover a pattern in a structured, semi-structured and unstructured data. It has a global impact on the technology as it is useful in AI, genetics, computer vision, business prediction, and others. In SDN, because of the logically centralized controller, a massive amount of information is generated and stored instantly. With the ever-increasing network information, machine learning techniques are formidable and play a vital role in discovering knowledge from the stored network information [3]–[5].

One of the most prominent challenges of the present world is energy since it has both economic and ecological issues. 10% of the global energy consumption is due to ICT sector out of which 2% is from network components. By 2020 the total electricity cost of cloud data centers is expected to increase by 63% [6], [7]. SDN enables us to achieve traffic proportional energy consumption through dynamic re-routing of flows. The practical solution is to sleep/turn off underutilized components during low traffic load. However,

there is a trade-off between performance and efficiency since turning off network components for sake of efficiency has an adverse effect on performance.

In our previous work [8], [9], we have proposed IP formulations and heuristics to maintain this trade-off. However, the efficiency of the three heuristics we proposed namely Next Shortest Path, Next Maximum Utility, and MEPT depend on the value of the utility interval parameters $Umin$ and $Umax$. In the previous work, we have determined these parameter values by brute force.

In this work, we propose MER-SDN, a framework that combines the capabilities of SDN and machine learning for energy efficient routing. MER-SDN is implemented on the POX controller. It extracts the topology and traffic information and stores it in a repository. It also uses PCA (Principal Component Analysis) for feature size reduction and linear regression for training the model. MER-SDN is a generic framework that can be applied to a range of energy efficient approaches. In this work, however, we use it to predict the optimal values of the $Umin$ and $Umax$ parameters for the MEPT heuristics. We also propose a heuristics to maximize the accuracy of the predicted $Umin$ and $Umax$ values.

The contributions of this work are as follows.

- We propose a three module machine learning framework for traffic proportional energy saving in SDN. The modules are Traffic Manager, Topology Manager, and Learning Machine.
- Most of the machine learning approaches used in SDN are for traffic classification, routing, intrusion detection, or attack prediction. To the best of our knowledge, we are the first in applying it to energy saving and performance combined.
- We present a full-fledged machine learning method that starts from feature extraction, applies feature reduction, and also provides a heuristics to increase the accuracy of the predictor to 100% in a constant time.
- We present the cross-fold validation results that show how we choose the number of principal components for PCA for the training set. The results indicate more than 65% feature size reduction.
- Our model predicts $Umin$ and $Umax$ with an accuracy of more than 70%. The refine heuristics we proposed to increase the accuracy converges to the optimal values with a speedup of 15 to 25 times as compared to the brute force approach.

The remainder of the paper is organized as follows. Section II presents related work. MER-SDN framework is discussed in section III. Section IV presents the experimental analysis of our approach. Conclusion and future work are discussed in section V

II. RELATED WORK

The use of machine learning techniques for energy efficiency in traditional networks has been studied [10], where the techniques are applied in assisting resource management, power distribution, demand forecasting, workload prediction, virtual machine placement prediction, memory assignment, CPU frequency, and traffic classification. The techniques range from supervised learning, unsupervised learning, reinforcement learning, and hybrid combination.

A meta-layered machine learning approach composed of multiple modules is proposed [3]. The goal of this approach is to mimic the results of heuristics used in traffic engineering to maximize the quality of service (QoS). However, each neural network per module is trained separately and each trained model operates separately for each demand pair. The drawback of this approach is that it does not represent the relationships between the demands.

Seer is a configurable platform for network intelligence based on SDN, Knowledge Centric Networking, and Big Data principles, where the goal is to accommodate the development of future algorithms and application that target network analytics [4]. It is also flexible in a sense that it allows high-level users to decide what network information to use for their goals. By focusing on reliability, the platform aspires to provide a scalable, fault-tolerant and real-time platform, of production quality.

Machine learning in SDN is also used in predicting the host to be attacked [11] using C4.5, Bayesian Network, Decision Table, and Naive-Bayes algorithms. Prediction of DDoS attack using neural network is implemented in NOX controller [12].

Another machine learning based approach NeuRoute is a dynamic framework which learns a routing algorithm and imitates its results using neural networks in real-time. NeuRoute is implemented on top of Google's TensorFlow machine learning framework and tested on POX controller. Experimental findings on GEANT topology show that the NeuRoute is faster than dynamic routing algorithms [5].

In contrast to the existing machine learning based solutions proposed for SDN, our framework models performance and energy efficiency at the same time. We present a method of representing traffic as features, perform feature size reduction using mathematically proven techniques, provide heuristics to increase the accuracy of the prediction to 100%. In our approach, we predict utility parameters $Umin$ and $Umax$ for the MEPT heuristics algorithm [9].

III. MER-SDN FRAMEWORK DESCRIPTION

We propose MER-SDN framework that utilizes machine learning techniques to achieve traffic proportional energy efficiency in SDN. The objectives are to jointly formulate energy efficiency and network performance, to propose generalized heuristics algorithms, and to apply machine learning approaches on SDN controller that learn from traffic, network and solution history.
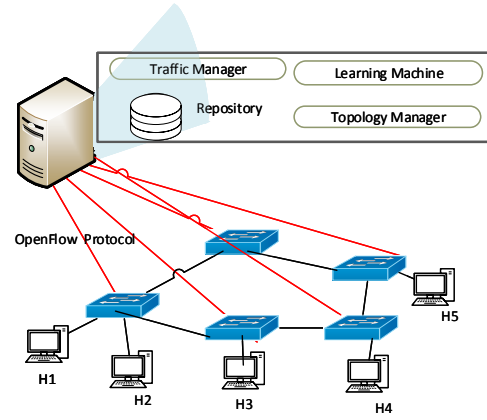

Fig. 1: MER-SDN: Machine Learning Framework for Energy Efficient Routing in Software Defined Networking

Figure 1 illustrates the MER-SDN framework consisting of three modules. The information of the traffic generated by the applications is passed to the traffic manager that stores details of traffic information in terms of source-destination pairs, rate, time a traffic demand arrives, and the total amount of flow. The status of the network and the topology information are fed to the learning machine from the switches, which generates an optimal sub-graph based on the traffic volume by learning from historical data. Since the module is designed to work in a dynamic environment, low traffic load would result in a sub-graph with a smaller number of active links and switches as compared to a subgraph in the case of high traffic load. The topology manager module is responsible for retrieving information about the organization and status of the network components. It also keeps track of cost information of links and forwarding switches. If a network component fails or is out of service, the topology manager updates the global topology information.

Figure 2 illustrates the machine learning stages applied to learn from traffic information and statistics of the network components. The pre-processing stage extracts features from the traffic, topology, switch, and link data and represents them using a matrix to perform size reduction. The training stage sets the hyper-parameters of the training model using cross-validation, and then builds a training model. The testing stage makes a prediction on the next sub-optimal graph that is proportional to the traffic volume. The refining prediction component improves the predicted state using a heuristics algorithm.

A. Feature Extraction

In machine learning, feature extraction is a technique used to select a subset of data more relevant to finding interesting patterns. Feature extraction involves feature representation and feature reduction. The performance of machine learning
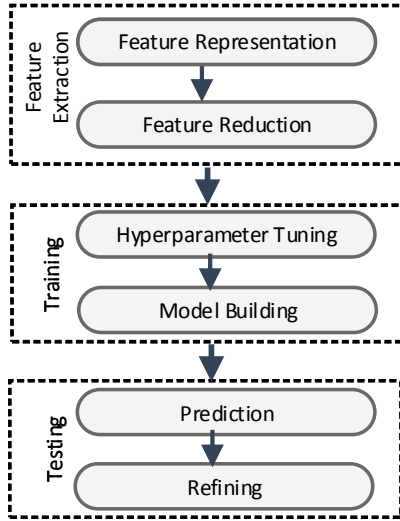
Fig. 2: Inside the Learning Machine

INPUT : Feature data X^{dxn} and k the number of principal components
OUTPUT : Feature data X^{dxk} where k is the number of principal components

1: $\bar{X} \leftarrow \sum_{i=1}^{n} X_i$ ▷ *mean of X*
2: $\bar{X} \leftarrow X - \bar{X}$ ▷ *mean normalize X*
3: $C \leftarrow \frac{1}{d}(X - \bar{X})^T(X\text{-}\bar{X})$ ▷ *C is the covariance matrix*
4: $V,E \leftarrow eig(C)$ ▷ *compute eigenvalues & eigenvectors*
5: $V \leftarrow \text{sort}_{desc}(V,E)$ ▷ *sort V based on E*
6: $W \leftarrow \text{eigenvecs}^k$ ▷ Projection matrix W^{d*k}
7: $X^{dxk} \leftarrow XW$ ▷ Project X on W space
8: **return** X^{dxk}, W

is at the first position while the eigenvector corresponding to the minimum eigenvalue is at the end of the list.

The next step in the PCA algorithm is to prepare the projection matrix W with the top k principal components. However, selecting the value of k is a significant step in PCA and challenging task. Small k value reduces the feature size significantly but preserves fewer information of the original data. The variance of the principal components shows the direction of the the maximum eigenvalue. The direction of the eigenvector corresponding the to maximum eigenvalue that carries most of the information in the original unreduced data. The larger the variance the more information the principal components carry. If we use the whole principal components the variance will be closer to 100%, and if the number of principal components chosen does not carry any information about the whole matrix, the value becomes 0. The variance decreases while moving from the first (largest) component to the last one. Line 6 computes the projection matrix W^{nxk}. W is used to project the original data X^{dxn} or a new data sample of various size. Line 7 reduces the dxn dimensional matrix X to dxk by projecting it over the eigenspace W. Line 8 returns the projected matrix X^{dxk} and the projection matrix W^{kn}.

B. Training

The training stage of MER-SDN has two parts: hyperparameter tuning and model training. We use linear regression to build our training model. However, setting the number of principal components needs to be investigated carefully. We use 10 fold cross-validation to pick parameter k. In 10-fold cross-validation, the original sample is randomly partitioned into 10 equal size subsamples. Of the 10 subsamples, a single subsample is retained as the validation data for testing the model, and the remaining 9 subsamples are used as training data. The cross-validation process is then repeated 10 times (the folds), with each of the 10 subsamples used exactly once as the validation data. The 10 results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and

methods depends on the choice of features. Complex features require memory, computational power, and longer training time. Moreover, the machine leaning algorithm over-fits the training set and generalizes poorly for unseen data. Feature reduction is a method of reducing the dimension of the feature set. Dimension reduction is the process of reducing the number of random variables under consideration by obtaining a set of principal variables [13]–[15]. Major techniques used in machine learning are Principal Component Analysis (PCA) [16], Factor Analysis (FA), Projection Pursuit (PP), and Independent Component Analysis (ICA) [17].

The network is represented as a directed graph where the nodes and the edges represent the switches and the links, respectively. A traffic flow is represented by the source node, destination node, and the flow rate. If the number of nodes in the network is N, then the size of the traffic matrix is Nx(N-1).

PCA is a linear combination of optimally-weighted observed variables. The outputs of PCA are these principal components whose numbers are less than or equal to the size of the original feature space. The principal components are orthogonal to each other. PCA is commonly used in face recognition, image classification, and unsupervised predictions. In this work, we use PCA to reduce the dimension of the feature set.

Algorithm 5 shows the steps used in PCA for feature size reduction. Line 1 computes the mean vector $\bar{X}$ of X. The dimension of $\bar{X}$ is equal to the feature size n. Line 2 mean normalizes the data. Mean normalization is necessary because it makes each feature component have same standard deviation which helps all principal components have equal weight. The next step in PCA is to compute the covariance matrix of the mean normalized data and compute the eigenvectors V and eigenvalues E as stipulated on lines 3 and 4 . Line 5 orders the V based on eigenvalues E in descending order. The eigenvector corresponding to the maximum eigenvalue

each observation is used for validation exactly once. This avoids bias in the training. After tuning the parameters, the next step is to build a regression model by using all the training dataset. The regression model is now ready to use for predicting new features.

C. Testing

Testing refers to applying the model trained to predict the values for unknown new data sample. First, we transform the test data set to eigenspace by a simple linear transformation. Then, we use the regression model we build (in section III-B) for prediction. However, in our experiments, the accuracy of the prediction is not 100% in constant time.

Algorithm 2 increases the accuracy of the model. The rationale behind the Refine algorithm is to increase the predicted $Umin$ by α until the energy saving decreases and to decrease the value of the predicted $Umax$ by α until the energy saving remains constant. And the outputs are the improved $Umin$ and $Umax$.

The inputs to the algorithm 2 are predicted $Umin_0$, predicted $Umax_0$, step size α, and threshold β. The step size parameter α is the value to add or subtract from $Umin$ and $Umax$ to find if the near values have better energy efficiency or not. The threshold parameter β is the terminating condition for the algorithm that measures that measures the energy saving difference between previous and current $Umin$ and $Umax$ values.

Lines 2, 3, and 4 calculate the efficiency of the energy saving algorithm MEPT for the $Umin$ parameter value of $Umin_0$, $Umin_0 - \alpha$, and $Umin_0 + \alpha$ respectively. Lines from 5 to 9 if the change in $Umin$ changes the energy saving. Line 11 sets the terminating condition for refining the value of the predicted $Umin$ by checking if the difference between the energy saving using the current $Umin$ and the previous $Umin$ is not greater than the threshold β. Lines from 13 to 15 alliteratively reduce the $Umax_0$ value till the energy saving is not changing according to line 17. The optimal value of $Umin$ and $Umax$ are found on lines 16, and 17.

IV. EXPERIMENTAL ANALYSIS

The experimental platform is based on POX controller and Mininet [18] network emulator installed on Ubuntu 16.04 64-bit. The topology is created on Mininet, and the heuristics are implemented on POX controller. Our experiments are conducted using real traces from SNDlib [19], in particular, the Abilene, GEANT, and Nobel-Germany dynamic traffic trace of the European research network. The metrics we use in this experiment are the accuracy of the predictor, feature size reduction due to PCA, cross-fold validation to pick the optimal number of principal components, speedup of the predictor and the refine algorithm as compared to the brute force method, energy efficiency, and average path length. Accuracy is calculated as $100*(1-\frac{|TV-PV|}{TV})\pm\epsilon$ where TV is the true value of the parameter, PV the predicted value of the parameter, and ϵ is the error tolerated. In this experiment, the value of ϵ is 3%. Speedup is calculated as $\frac{100}{N}$ where N

Algorithm 2 Refine: Improves the predicted parameters $Umin_0$ and $Umax_0$ values for better efficiency

INPUT : Predicted parameters $Umin_0$ and $Umax_0$, change α, threshold β.

OUTPUT : Improved parameters $Umin$, $Umax$

```
1: repeat
2:     EE_curr ← EE(Umin_0, Umax_0)
3:     EE_prev ← EE(Umin_0 - α, Umax_0)
4:     EE_next ← EE(Umin_0 + α, Umax_0)
5:     if EE_prev < EE_next then
6:         Umin_0 ← Umin_0 + α
7:     else
8:         Umin_0 ← Umin_0 - α
9:     end if
10:     EE_new ← EE(Umin_0, Umax_0)
11: until ABS(EE_curr - EE_new) ≤ β
12: while EE_curr ≥ EE(Umin_0, Umax_0 - α) do
13:     Umax_0 ← Umax_0 - α
14:     EE_curr ← EE(Umin_0, Umax_0)
15: end while
16: Umin ← Umin_0
17: Umax ← Umax_0
18: return Umin, Umax
```

TABLE I: Topologies and Traces

Topology	Nodes	Edges	Avg Deg.	Feature Size	Snapshot Minutes
Abilene	12	15	2.5	132	5
GEANT	22	36	4.35	462	15
Nobel-Germany	17	26	3.06	272	5

is the number of times the energy saving algorithm (MEPT) is run before the Refine algorithm gets the optimal value.

Table I presents the topologies, traffic characteristic and the size of the features used in this experiment. The features are extracted from a snapshot of the network aggregated in 5 to 15 minutes. The features are represented as a matrix where each row is an N(N-1) vector representing the rates between a source and destination pairs. For the Abilene topology with 12 nodes, accordingly, the dimension of the feature is

---

Algorithm 3 PCA: Reduce the traffic matrix X^{nxd} to X^{nxk} and produce the projection matrix W^{dxk}

Input: Traffic matrix X^{nxd} and k the number of principal components

Output: Feature data X^{nxk} and projection matrix W^{dxk} where k is the number of principal components and d is

```
1: X̄ ← Σ_{i=1}^{d} X_i              ▷ mean of X
2: T ← X - X̄                        ▷ mean normalize T
3: C ← (1/n)(X - X̄)^T(X - X̄)  ▷ C is the covariance matrix
4: V, E ← eig(C)        ▷ computer eigen value and vector
5: V ← sort_desc(V, E)            ▷ sort V based on E
6: W ← eigenvecs^k            ▷ Projection matrix W^{dxk}
7: X^{nxk} ← XW               ▷ Project X on W space
```

12*11=132. Accordingly, the feature sizes for GEANT and Nobel-Germany are 462 and 272 respectively. We train the models for traffic size of 10% to 90%.

Algorithm 4 MaxRESDN ($\mathbb{G}$, $\mathbb{F}$, $\mathbb{U}$, $Umin$, $Umax$)

1: **Input:** Graph $\mathbb{G}$, set of traffic flow $\mathbb{F}$, utility of links $\mathbb{U}$, minimum utility $Umin$, and maximum utility $Umax$
2: **Output:** Modified utility of links $\mathbb{U}$ and graph $\mathbb{G}$
3: **for all** $f = (sr, ds, \lambda_f) \in F$ **do**
4:    $path_f \leftarrow$ `PathMaxRESDN` (sr,ds,λ_f)
5:    **for all** $e_{ij} \in path_f$ **do**
6:       $U_{ij} \leftarrow U_{ij} + \dfrac{\lambda_f}{W_{ij}}$
7:    **end for**
8: **end for**
9: **for all** $e_{ij} \in \mathbb{E}$ **do**
10:    **if** $U_{ij} == 0$ **then**
11:       $L_{ij} \leftarrow 0$
12:    **end if**
13: **end for**

Algorithm 5 PCA: Reduce the traffic matrix $\mathbf{X}^{nxd}$ to $\mathbf{X}^{nxk}$ and produce the projection matrix $\mathbf{W}^{dxk}$

Input : Traffic matrix $\mathbf{X}^{nxd}$ and k the number of principal components
Output : Feature data $\mathbf{X}^{nxk}$ and projection matrix $\mathbf{W}^{dxk}$ where k is the number of principal components and d is

1: $\bar{X} \leftarrow \sum\limits_{i=1}^{d} X_i$ ▷ *mean of X*
2: $T \leftarrow \mathbf{X} - \bar{X}$ ▷ mean normalize T
3: $C \leftarrow \frac{1}{n}(X - \bar{X})^T (X - \bar{X})$ ▷ *C is the covariance matrix*
4: $V, E \leftarrow eig(C)$ ▷ computer eigen value and vector
5: $V \leftarrow sort_{desc}(V, E)$ ▷ sort V based on E
6: $W \leftarrow eigenvecs^k$ ▷ Projection matrix W^{dxk}
7: $X^{nxk} \leftarrow XW$ ▷ Project X on W space

Figure 3 show size reduction, cross-validation accuracy and variance of the model for the Abilene, GEANT, and Nobel-Germany topology and trace. Size reduction is inversely proportional to accuracy and variance. The percentage of principal components we picked from the 10-fold cross validation are 30%, 32% and 35% which correspond to accuracy values 78%, 79%, 80% and feature size reduction of 70%, 68%, 65% for the Abilene, GEANT and Nobel-Germany topologies and traces. The value of k we have chosen for the PCA is 40, 148, and 96 for the three topologies and traces. The number of principal components we picked is up to 5% larger than where the accuracy and the size reduction plots intersect. The choice is carefully made so that the model would not over-fit the data at the same time contain at least 80% of the information in the original data.

Figure 4 shows the accuracy of predicting $Umin$, $Umax$, $Umin/Umax$ ($Umax$ given $Umin$ is known), $Umax/Umin$ ($Umin$ given $Umax$ is known) versus for the GEANT data set. The accuracy of predicting $Umin$ ranges between 68% to 75%. For the Abilene trace, $Umax$

(a) Abilene
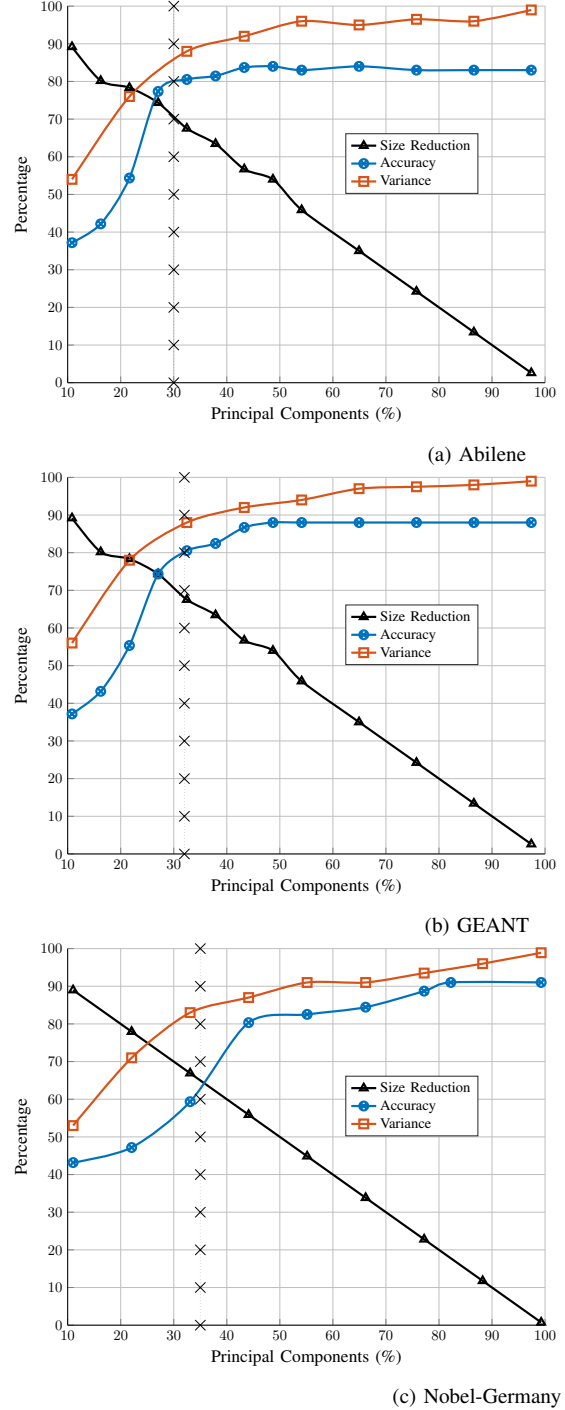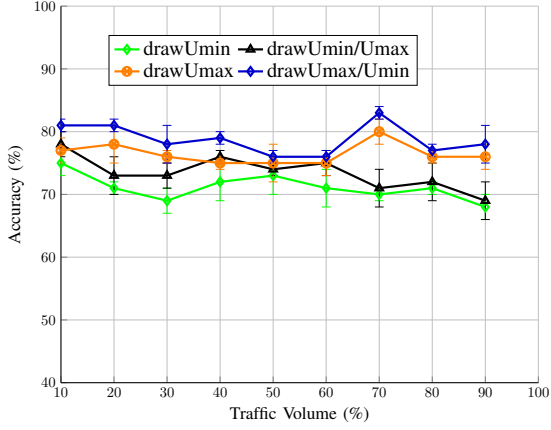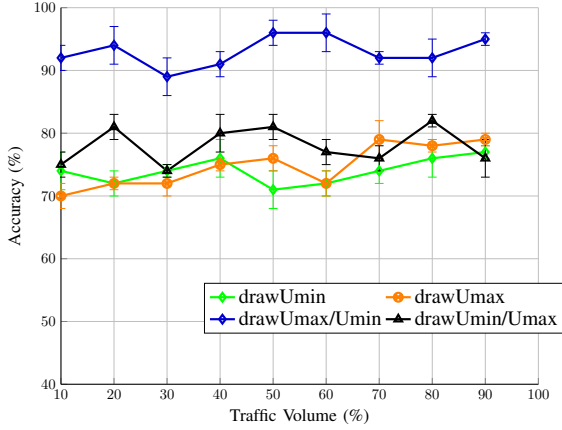

(b) GEANT

(c) Nobel-Germany

Fig. 3: Cross validation results of the percentage of PCs versus feature size reduction, accuracy, and variance for Abilene, GEANT, and Nobel-Germany topology traces. It also shows the optimal percentage of PCs selected for each topology and trace.
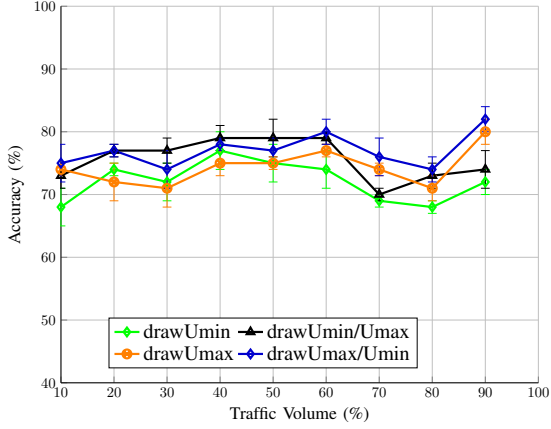
prediction accuracy is 3 to 5% better than $Umin$'s prediction. An interesting observation from this experiment is that the accuracy of $Umin$ and $Umax$ increase if $Umax$ and $Umin$ are known apriori. In case of the GEANT topology trace, a prior knowledge of $Umin$ increases the prediction accuracy of $Umax$ by at least 15%. The accuracy of the predictor is independent of the traffic volume.
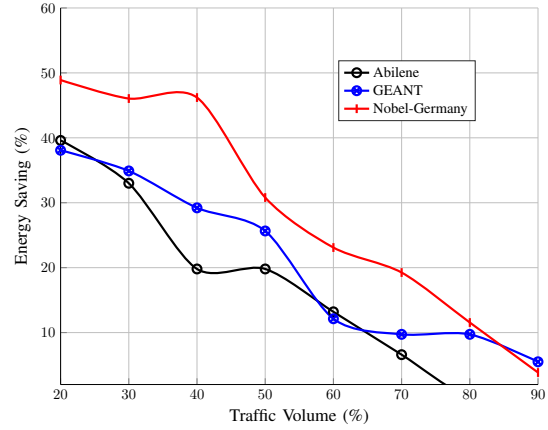
(a) Abilene

(b) GEANT

(c) Nobel-Germany

Fig. 4: Accuracy for predicting $Umin$, $Umax$, $Umin/Umax$ ($Umin$ given $Umax$ is known, and $Umin/Umax$ ($Umax$ given $Umin$ is known a) Abilene b) GEANT and c) Nobel-Germany topology traces

Table II shows fast the Refine algorithm converges the optimal values of $Umin$ and $Umax$ parameters relative to the brute force method. The brute force method checks all values from 0% to 100 % and selects the optimal $Umin$ and $Umax$ for highest energy saving. Since the accuracy of the prediction is not 100%, the Refine heuristic improves the predicted values to reach the optimal value with few numbers
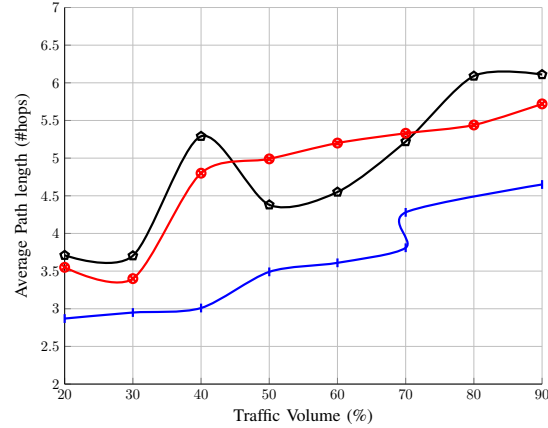
TABLE II: Refine algorithm speedup for convergence of the $Umin$ and $Umax$ parameters of MEPT heuristics algorithm as compared to the brute force method

Traffic	Abiline		GEANT		Nobel-Germany	
	$Umin$	$Umax$	$Umin$	$Umax$	$Umin$	$Umax$
10	17.86	21.43	18.37	15.43	14.29	18.37
20	16.07	22.69	16.77	16.77	18.37	16.77
30	14.84	20.3	18.37	16.77	16.77	16.07
40	16.77	19.29	20.3	19.29	21.43	19.29
50	17.53	19.29	16.07	20.3	19.29	19.29
60	16.07	19.29	16.77	16.77	18.37	21.43
70	15.43	25.71	18.37	24.11	14.84	18.37
80	16.07	20.3	20.3	22.69	14.29	16.07
90	14.29	20.3	21.43	24.11	16.77	25.71

of steps. The speedup for traffic ranging from 10% to 90% traffic volume is 14.85X to 25.71X the brute force method. Like the accuracy of the predictor, the speedup of the Refine heuristics is independent of the traffic volume.

(a) Energy Saving

(b) Average Path Length

Fig. 5: Energy Efficiency and Performance of the MEPT heuristics according to the predicted $Umin$ and $Umax$ values for the Abilene, GEANT, and Nobel-Germany topology traces

Figure 5a and 5b illustrates the energy efficiency and average path length of the MEPT heuristics according to the predicted $Umin$ and $Umax$ parameters after applying the Refine heuristics. Results show that MEPT achieves an energy saving of 48% for low traffic. The trends in energy saving are indirectly proportional to the traffic volume. This

shows MEPT algorithm makes traffic proportional energy saving. The average path length measurement shows that increase in traffic increases the average path length. This is because as the volume of traffic increases, the shortest paths become overloaded, because of flows have to be re-routed to longer paths.

The significance of the machine learning method lies in the fact that it predicts $Umin$ and $Umax$ for a given new traffic. Getting the optimal values the parameters increases the EPT value. Maximum EPT value increases utilities of links, energy saving, and also maintains an acceptable network performance.

V. Conclusion and Future Work

SDN is a very powerful networking paradigm that allows flexibility in the control and management of a network through re-routing flows in order to achieve efficiency, performance, load balancing, and security. In this work, we proposed MER-SDN, a machine learning based framework for traffic aware energy efficient routing in SDN. We used topology and traffic as features to train our model. We also employed PCA and achieved a feature size reduction of more than 65% on real-world network topology and dynamic traffic traces. Particularly, we tested MER-SDN to predict $Umin$ and $Umax$ parameters for the energy efficient MEPT heuristics algorithm. In addition to the prediction, we also proposed a heuristics to increase the accuracy of the predictor to 100%. Experiment results show that the accuracy of predicting $Umin$ and $Umax$ are more than 70%. The refining heuristics algorithm converges to the optimal $Umin$ and $Umax$ values 15 to 25 times faster than the brute force method.

Our framework is tested by taking snapshots of historical traffic traces. As future work, we plan to incorporate time dimension to the features. The effect of the traffic volume on the accuracy of our model would also be discussed. We aim at using reinforcement learning technique to help the controller to be self and incrementally learn and predict in a dynamic environment. We also plan to add link and switch status as a feature, and conduct comprehensive experiments on the effect of the energy saving approach on throughput and delay.

References

[1] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *Communications Surveys and Tutorials, IEEE*, vol. 16, no. 3, pp. 1617–1634, 2014.

[2] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[3] L. Yanjun, L. Xiaobo, and Y. Osamu, "Traffic engineering framework with machine learning based meta-layer in software-defined networks," in *Network Infrastructure and Digital Content (IC-NIDC), 2014 4th IEEE International Conference on*. IEEE, 2014, pp. 121–125.

[4] K. Sideris, R. Nejabati, and D. Simeonidou, "Seer: Empowering software defined networking with data analytics," in *2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS)*, Dec 2016, pp. 181–188.

[5] A. Azzouni, R. Boutaba, and G. Pujolle, "Neuroute: Predictive dynamic routing for software-defined networks," *CoRR*, vol. abs/1709.06002, 2017.

[6] R. Maaloul, L. Chaari, and B. Cousin, "Energy saving in carrier-grade networks: A survey," *Computer Standards and Interfaces*, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0920548916301817

[7] GreenPeace. (2014) Clicking clean: How the Companies are creating the green internet. [Online]. Available: http://www.greenpeace.org/international/en/

[8] B. G. Assefa and O. Ozkasap, "Link utility and traffic aware energy saving in software defined networks," *2017 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*.

[9] ——, "Framework for traffic proportionalenergy efficiency in software defined networks," *2018 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)-preprint*.

[10] M. Demirci, "A survey of machine learning applications for energy-efficient resource management in cloud computing environments," in *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*. IEEE, 2015, pp. 1185–1190.

[11] S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa, and B. Yang, "Predicting network attack patterns in sdn using machine learning approach," in *Network Function Virtualization and Software Defined Networks (NFV-SDN), IEEE Conference on*. IEEE, 2016, pp. 167–172.

[12] J. Ashraf and S. Latif, "Handling intrusion and ddos attacks in software defined networks using machine learning techniques," in *Software Engineering Conference (NSEC), 2014 National*. IEEE, 2014, pp. 55–60.

[13] P. Pudil and J. Hovovicova, "Novel methods for subset selection with respect to problem knowledge," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 2, pp. 66–74, 1998.

[14] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[15] H. Samet, *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.

[16] N. Kambhatla and T. K. Leen, "Dimension reduction by local principal component analysis," *Dimension*, vol. 9, no. 7, 2006.

[17] I. K. Fodor, "A survey of dimension reduction techniques," Lawrence Livermore National Lab., CA (US), Tech. Rep., 2002.

[18] B. Lantz, B. Heller, and 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*, 2010, p. 19.

[19] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *Proceedings of the 3rd International Network Optimization Conference (INOC 2007)*.