Evaluation of Topology Optimization Objectives in IP Networks

Y. Sinan Hanay, Shin'ichi Arakawa, and Masayuki Murata

Abstract: **In the past, various optimization objective functions have been proposed to help in network optimization, especially for use in traffic engineering (TE) and topology optimization. This variety of optimization objectives resulted in the emergence of algorithms targeting different objectives. However, the role of the objective function has been largely overlooked. Because, the choice of a particular objective function was not justified in most of the cases. Some researchers criticized this arbitrary selection of objective functions. Even though some researchers intuitively suggest using a specific objective, only few work tackled with the problem of evaluating the objectives. In this paper, we evaluate various network optimization objectives on topology optimization. Previously, a study analyzed the efficiency of some routing optimization objectives using linear programming (LP) by linear relaxation. However, some of the objective functions are nonlinear, and such a linear relaxation does not treat each objective equally.The difficulty arises due to the fact that optimization algorithms are objective function tailored heuristics. To achieve fairness, we compare and analyze different traffic optimization objectives for topology optimization using neural networks which are used to model nonlinear relations. By using neural networks, we strive to avoid any unfairness, such as obviating linear approximation. Also, our work suggests which features are meaningful for machine learning in network optimization. Our method partially agrees with the previous work, and we conclude that delay is the best performing optimization objective.**

Index Terms: **Machine learning, network optimization, neural networks, optimization objectives, topology optimization.**

I. INTRODUCTION

TRAFFIC engineering (TE) and topology optimization are two domains in network optimization. Considerable research attention has been devoted to developing novel methods for both optimization problems. TE focuses on routing optimization and load balancing; in a way, TE directs the traffic to feasible paths. On the other hand, topology optimization focuses on finding topologies that can accommodate the traffic. Nevertheless, both domains attempt to optimize an objective function such as minimizing maximum link utilization, average delay, weighted hop count, average queuing delay or maximizing available bandwidth. These are some well-known network-wide optimization objectives.

Manuscript received June 23, 2017; approved for publication by Jeongyeup Pa, Division III Editor, November 7, 2018.

Y. S. Hanay is with Erzurum Technical University, Turkey, email:sinan.hanay@erzurum.edu.tr.

S. Arakawa and M. Murata are with Osaka University, e-mail:{arakawa, murata}@ist.osaka-u.ac.jp.

Preliminary results of this work have been published in IEEE Conference on Local Computer Networks 2015 [1].

Digital Object Identifier: 10.1109/JCN.2019.000014

Routing protocols use shortest path algorithms, and it is possible to adapt the protocol to target any objective function by changing only link weights [2]. Most of the researches in routing optimization have focused on optimizing the weights to achieve an optimization objective [2], [3]. On the other hand, little has been done on the evaluation of how well optimization objectives do, such as in [4]. In that pioneering work, the researchers investigated the efficiency of different optimization objectives. They took the linear programming (LP) approach for evaluating different objectives even as making some linear approximations on nonlinear optimization objectives. Rightfully, they acknowledge the shortcomings of linear approximations. As a result, there remains a need for a fair comparison of different optimization objectives.

In this work, we evaluate different optimization objectives in topology optimization problem in optical networks, the core infrastructure of the Internet today. Optical fiber has become the main choice of communication medium for long-haul networks replacing relatively lossy copper wire medium. This migration has allowed transmitting higher bandwidths of data with fewer repeaters over long distances. In turn, the Internet witnessed a thousand-fold increase in bandwidths during the 1990s [5]. In addition, optical communication is capable of carrying many channels simultaneously using wavelength-division multiplexing (WDM).These capabilities of the optical medium allow establishing of many different *virtual topologies* on top of the very same physical topology. Selecting an efficient virtual topology (VT) is an important problem in autonomous systems (AS).

Since the mid-1990s, there has been a great effort on VT optimization. Researchers commonly used maximum link utilization, delay or average weighted hop as objectives [6]. In this study, we evaluate the most common, nonparametric objective functions for VT optimization problem. However, the main contribution of our work is to provide a fair comparison for different optimization objectives. We strive to provide fair comparison by using a machine learning algorithm. Previously, nonlinear objective functions were evaluated using linear approximations [4], [2]. On the other hand, with *neural networks*, such unfairness in the evaluation can be avoided. More importantly, we evaluate those objectives under realistic, dynamic traffic. This allows us to make our conclusions more comprehensive. We conclude that some optimization objectives are better to target than others. We observed how a very commonly used objective function results in poor performance. Our results regarding the best optimization objective agrees with the previous work [4].

The remainder of this paper is organized as follows. Section II presents the motivation for this work, Section III presents preliminaries related to the neural networks algorithm we use. Section IV describes the optimization objectives we cover. Sec-

tion V presents the simulation results, the related work is presented in Section VI, and Section VII concludes the paper.

II. MOTIVATION

Although there has been an enormous effort concentrated on TE and topology optimization, there has been scant attempt to understand how well the optimization objectives perform. The optimization objective is a vital aspect of any work tackling with network optimization. Few researchers have addressed the problem of analyzing different objectives. There has been only a single study that considered different TE optimization objectives and the authors concluded that some objective functions are more worthy than others [4]. However, as the authors acknowledge, the use of linear approximation of nonlinear objective functions limits the scope of conclusion, and an alternative approach is crucial to understand the performance of different optimization objectives.

To illustrate why the selection of objective function matters, consider the following optimization objective: One rule of thumb is to maintain link utilizations under 50 percent for all links or to minimize maximum link utilization. However, minimizing maximum link utilization is overly sensitive to bottleneck links [4], [2]. That is, maximum link utilization is a very local metric, which may be far from capturing the global network performance. Another way to look at this is to observe that only one link determines the objective value [3]. For example, two solutions with same maximum utilization but a different mean utilization will have the same value. This is a common problem with objective functions based on min-max formulations.

Even though the maximum link utilization objective has such a shortcoming, nonetheless it has been one of the most common optimization metric in topology optimization studies [6]. For example, some recent high impact studies on software-defined networks (SDN) used maximum utilization [7], [8]. In this work, we strive to achieve a fair comparison through a machine learning algorithm in topology optimization setting.

A. Topology Optimization

Topology optimization takes place at the physical layer at the core of the Internet. Fig. 1 illustrates the topology optimization problem. In an all-optical, IP-over-WDM network, each router is equipped with a set of transmitters and receivers. Each fiber link can carry a certain number of wavelengths. Optical cross-connects serve as a switching device for optical signals, and associates and incoming link with an outgoing link. This allows the possibility of establishing various *virtual topologies* on top of a physical topology. Finding efficient virtual topologies is the problem. This topology optimization problem is more specifically referred as virtual topology design (VTD) to signify the fact that underlying physical topology does not change (i.e. fiber links). The problem is called virtual topology reconfiguration (VTR), when the virtual topology is updated periodically. In this work, we evaluate objectives in VTR setting, which recent research has focused on.

Fig. 1 illustrates the topology optimization problem. In the illustration, each fiber link can carry two wavelengths. Wavelength 1 connects 1-hop nodes (i.e. A-B, B-D and C-D), and Wavelength 2 connects pair B-C. The path between B and C is seamless from the IP layer perspective. These seamless paths connecting nodes in IP layer called "lightpaths". Note that in this example, there are many possible assignments even with 2 wavelengths. For example, A and D could be connected using wavelength 1, in which case there would be no lightpath between A and B, and between B and D. VTR problem is to find the topology that satisfies some optimization goal, thus it is the optimization objective that determines the final virtual topology.

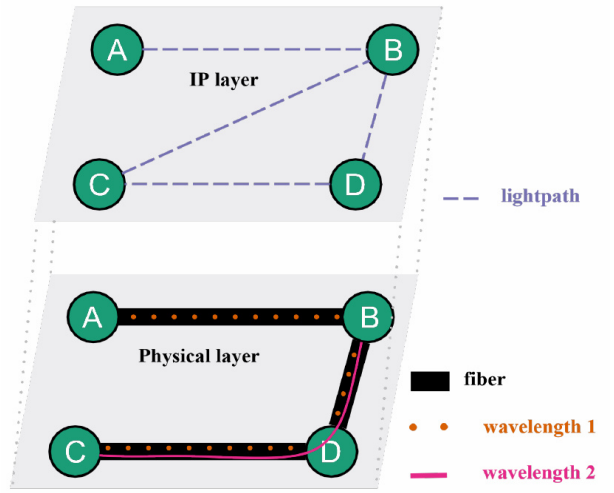


Fig. 1. Illustration of topology optimization problem in IP over WDM networks. A physical topology with 4 routing nodes (A, B, C and D) and the corresponding virtual topology at the IP layer is shown.

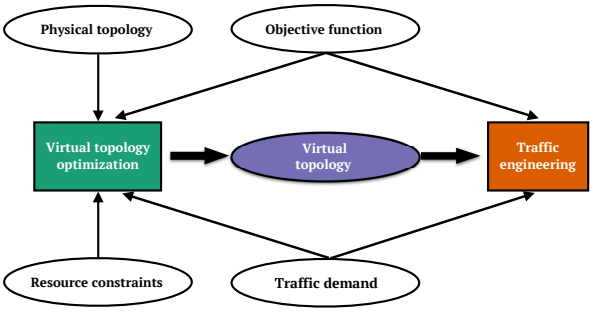


Fig. 2. The flowchart shows the relation between virtual topology design and traffic engineering. The physical topology is the optical network, and the number of wavelengths and ports are resource constraints.

Figure 2 illustrates the relation between TE and VTD/VTR. Traffic demand is same for both problems, however, the objective function may differ. Traditionally, VTR and routing problems (i.e. TE) are treated separately [6]. Despite our efforts to combine both problems, the simulation results of this intricate problem prevented us from drawing any meaningful conclusion. Thus, we limit our focus on topology optimization, and use unit cost routing where each link has the same weight.

B. Why Machine Learning?

In recent years, there has been several influential work in networking community using machine learning as diverse as con-

gestion control [9], traffic classification [10] and intrusion detection [11]. The researchers demonstrated that several important problems can be solved efficiently with machine learning.

In our case, we use machine learning due to limitations. Most of the VTR optimization methods use heuristics. For each optimization, it is necessary to use a different heuristic. Using different heuristics prevents comparison of different objectives on fair grounds. LP formulations use linear approximations on nonlinear objectives, and heuristics are tailored to the objective functions. Thus, for this problem, the use of machine learning is rather a necessity than a choice.

Before delving into the optimization problem, we need to show that the VTR problem can benefit from machine learning. Machine learning works well when there is a trend or pattern in the data or variables that can be captured statistically. It has been long discussed whether Internet traffic is self-similar after the pioneering work of Leland *et al.* [12]. There is no consensus on self similarity of traffic in the research community. However, long range dependency (LRD) was accepted and observed in real traffic settings [13]–[15]. The next section presents the preliminaries and the algorithm.

III. PRELIMINARIES

The machine learning algorithm we use in this work is called Attractor Selection Based (ASB) topology control. To understand the outcomes of this work, it is not essential to understand ASB fully. However, we briefly review it. The details of ASB can be found in prior work [16].

ASB is built on neural networks. The learning type it uses can be regarded as reinforcement learning in a broad sense. Unlike supervised learning, in reinforcement learning there is no training dataset. In reinforcement learning, the learner takes actions, and upon the consequences of its action, the learner modifies its knowledge. ASB begins randomly exploring viable topologies. Upon discovering a "good topology", which is explained in detail in Section IV.C, ASB stores it in a memory. Those stored good topologies serve as a guide for ASB in future explorations by attracting the algorithm to converge a topology similar to themselves. Since those topologies attract the state towards themselves, they are called *attractors*.

To give a sense of problem size, we walk through the numbers. For N nodes, there are $N \times (N-1)$ node pairs. Instead of an adjacency matrix, we describe topologies by bit vectors. Bit vectors are necessary as the neural network we use works only with them. Hence, a bit-vector of size $N \times (N-1)$ can represent any topology, and total number of possible topologies are $2^{N\times(N-1)}$. However, due to resource constraints such as number of transmitters/receivers at each node, it is impossible to establish all possible topologies.

A. Neural Memories

Once ASB algorithm finds a good topology, it stores the topology inside a neural memory. Neural memories are different than traditional computer memories, in the sense that they do not store information as it is. In computer memories, the stored elements are read from bit cells. On the other hand, in neural memories the output is provided through a mathematical operation, such as a matrix multiplication.

The type of the neural memory we use in this work is an auto-associative memory. Auto-associative memories can be used to correct noisy inputs by trying to associate a given input to one of the stored patterns (e.g., topologies). Auto-associative memories resemble content addressable memories (CAMs), the values are supplied instead of addresses. Unlike CAMs, auto-associative memories return values. If a query is not stored in the memory, then associative memory returns the closest element to the query.

Consider the following neural memory. The output $\mathcal{O}$ for an input $\mathcal{I}$ is calculated by

$$\mathcal{O} = sgn(\mathcal{I}\,\mathcal{W}). \tag{1}$$

Here $\mathcal{O}$ and $\mathcal{I}$ are row vectors, $sgn()$ is the sign function given by

$$sgn(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{2}$$

and the weight matrix $\mathcal{W}$ can be calculated as

$$\mathcal{W} = \mathcal{A}^\top \mathcal{A}, \tag{3}$$

where $\mathcal{A}$ represents the stored topologies (i.e. each row of $\mathcal{A}$ is a topology).

It is more efficient to encode values as bipolar (i.e., $[1, -1]$) rather than $[1, 0]$ in neural memories [17]. We proceed with a toy example. For example, consider storing of two 4-bit values, such as:

$$\mathcal{A} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & -1 & 1 & -1 \end{bmatrix}$$

and using (3), $\mathcal{W}$ can be calculated as

$$\mathcal{W} = \begin{bmatrix} 2 & 0 & -2 & 2 \\ 0 & 2 & 0 & 0 \\ -2 & 0 & 2 & -2 \\ 2 & 0 & -2 & 2 \end{bmatrix}$$

Using (1), it is straightforward to check the following equalities hold

$$\text{sgn}(\mathcal{A}_1\,\mathcal{W}) = \mathcal{A}_1$$
$$\text{sgn}(\mathcal{A}_2\,\mathcal{W}) = \mathcal{A}_2.$$

This means, when a stored input is provided, the memory returns the stored input. However, the main function of auto-associative memories is to associate noisy inputs to stored values. For example, let $\widetilde{\mathcal{A}}_1$ denote a noisy version of $\mathcal{A}_\infty$, where the last bit is flipped, that is $\widetilde{\mathcal{A}}_1 = [1 \quad -1 \quad -1 \quad -1]$. This noisy input can be corrected as the following holds:

$$sgn(\widetilde{\mathcal{A}}_1\,W) = \mathcal{A}_1.$$

In some cases, the noisy inputs cannot be corrected. For example, an input where the second bit of $\mathcal{A}_2$ is flipped, that is $\widetilde{\mathcal{A}}_2 = [-1 \quad 1 \quad 1 \quad -1]$ cannot be corrected. The memory returns $[-1 \quad 1 \quad 1 \quad -1]$, which is not $\mathcal{A}_2$.

We have just shown one way of constructing an auto-associative memory, based on Hebbian learning (i.e. autocorrelation matrix) [18]. More efficient memories can be constructed using different weight matrices, such as using multistate memories [19], [20].

```
1:  procedure ASB( time t )
2:      Monitor current topology T_i
3:      Calculate performance metric α
4:
5:      if α_{t-1} < α_th  and  α_t > α_th  then
6:          remove the oldest topology in A                ▷ FIFO
7:          store T_i in A using storeTopology(A, T_⟩)
8:
9:      Calculate x_i for each node-pair by Equation 4
10:
11:     for all Node pair i do
12:         if x_i > 0.5 and resources available then
13:             Establish path i
14:         else if x_i < 0.5 then
15:             Terminate path i and free resources
16:
17: procedure STORETOPOLOGY(A, Topology)
18:     return W ← A^⊤ A
```

Fig. 3. ASB algorithm.

B. ASB Algorithm

ASB aims to find an optimal virtual topology (VT), and it changes topologies according to the following equation [16]:

$$\frac{dx_i}{dt} = \underbrace{\left[f\left(\sum_{j=1}^{n} w_{ij} x_j \right) - x_i \right]}_{\text{auto-associative memory}} \alpha \; + \; \underbrace{\mathcal{N}(0,1)}_{\text{random walk}}, \qquad (4)$$

where $\mathcal{N}(0,1)$ is the standard normal random variable, f can be sign or sigmoid function. The value α is the optimization metric to be maximized. For example, if we want to minimize u_{max}, then α should be a decreasing function of u_{max}, since they are inversely related. Here, x_i represents the likelihood of establishing a path for node pair i. In each round, ASB makes changes to the present topology based on x_i values. For example, if x_i is greater than 0.5, then lightpath for pair i is established; otherwise the lightpath is terminated (if it exists). Of course, the lightpath is established only if corresponding resources are available (i.e. wavelength and ports).

As (4) shows, ASB has two components: auto-associative memory and random walk. When the topology performs well (i.e. α is high), the auto-associative memory part steers the topology selection towards previously stored topologies. On the other hand, when α is low; ASB reduces the contribution of stored topologies.

Fig. 3 gives an overview of ASB algorithm. First, ASB monitors the performance (line 2). Then, it converts this u_{max} value to an optimization objective parameter α (line 3). Then it is time to check if the current topology is a good topology worth to remember (line 5). The current value of α is checked against a threshold value α_{th}, which is 0.5. Then it checks the performance at the last round α_{t-1} to threshold. This is done to determine if the topology is in a transition from a bad topology to good topology. This also prevents possible hysteresis. If, the conditions to add a new topology to attractor list is satisfied, then the current topology is added as a new attractor in a FIFO fashion (Line 6). Regardless of whether present topology is an attractor or not, ASB updates topology, each round by (4) as line 7 shows.

In line 9, the algorithm calculates the expression levels, which determines which paths needs to be established based on the value. If the expression level for a path is more than 0.5, then the corresponding path will be established. Note here that, the paths can be established as long as physical resources allow (i.e. availability of wavelengths, ports) (line 12). If the expression level is less than 0.5, and if that path already exists, then that path is terminated and the attached physical resources are freed (line 15).

Before we start our evaluation, we need to check if the topology optimization problem is suitable for machine learning. In the next section, we analyze real network traffic traces to check the feasibility of using machine learning and also discuss why machine learning is a good candidate for this problem.

IV. COMPARISON OF OPTIMIZATION OBJECTIVES

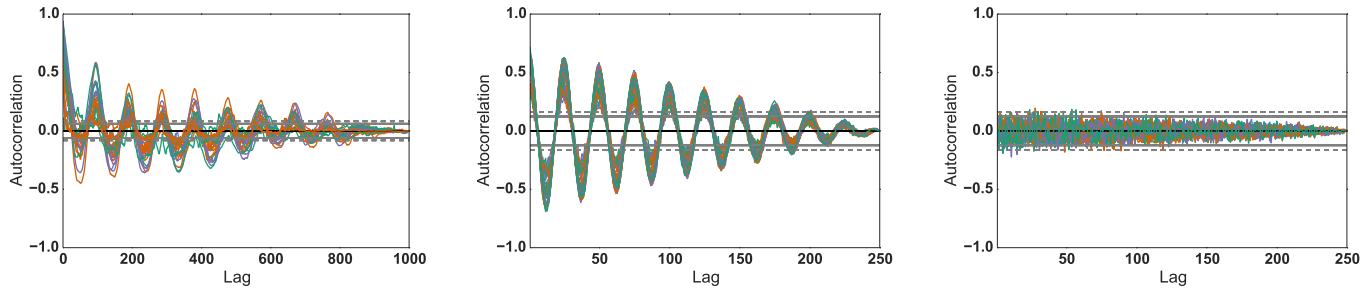
In this section, first we discuss why machine learning can be a good candidate. After that, we will check whether the use of machine learning is feasible through analyzing real network traffic traces. Finally, we look at the commonly used optimization metrics.

A discussion of traffic engineering performance objectives can be found in RFC 2702 [21]. In Section 2.1 of RFC 2702, the performance objectives are classified into two groups as traffic oriented and resource oriented objectives. For traffic oriented objectives, four objectives are presented: minimization of packet loss, minimization of delay, maximization of throughput and enforcement of service-level agreements. In this work, we use two of these four objectives: minimization of delay and maximization of throughput. As traffic load-level simulators are used in topology optimization research, it is infeasible to trace packet loss. However, we use a objective that aims to minimize the utilization on maximally loaded link, and it is directly correlated with packet losses. In addition to these objectives, we included two more objectives in this study: 1) minimizing the average weighted number of hops, and 2) minimizing the variance of link utilizations.

In this work, we evaluate network-wide metrics only, as this is a network-wide optimization problem. It is impossible to evaluate end host based metrics such as average flow completion time, because the network operator cannot have such knowledge. The metrics we discuss in this section are the most common metrics used in traffic engineering and topology optimization studies. We focus on nonparametric functions, as parametric functions require parameter space search. Even though Balon et al. analyzed some parametric objectives, they stated their preference for nonparametric objectives [4]. This work also gives an insight on what features are useful in optimizing networks using machine learning

A. Traffic Analysis

First, we analyze real traffic trace from GEANT topology, which consists of 23 nodes, provided by the TOTEM



(a) (b) (c)

Fig. 4. Autocorrelation function plots (ACF) of traffic loads for different source-destination pairs are shown. 95 and 99 percent confidence intervals are shown as dashes and lines: (a) Real traffic trace from GEANT topology, (b) synthetic traffic used in this work, and (c) random traffic.

project [22]. Figure 4 shows the autocorrelations of the traffic loads between each pair in the traffic data taken from GEANT topology between January 1st to January 11th 2006. A signal is correlated, if the autocorrelation function (ACF) coefficients are strictly non-zero. Note that each consecutive point in GEANT figure corresponds to a 15 minute interval, whereas in synthetic traffic and random traffic, it is equal to an hour interval. In the figure, the autocorrelations of the traffic load from nodes 13 and 20 to all other destination nodes are plotted. In the x-axis, each interval corresponds a 15 minute sampling interval. Thus, for example, $x = 96$ corresponds to one day. The figure reveals that at about every 96 rounds, there is a strong dependency with the previous traffic. Real intra-domain network traffic shows a trend, that can be captured by a machine learning algorithm. Therefore, learning is feasible for this problem. The figure clearly indicates a strong dependence with a period of about 24 hours. After each day, the correlation reduces. After 9 days, the correlation falls between the confidence intervals, which means the traffic is no longer correlated.

ACF plots are good to give a general sense of correlation of signals. To understand the correlation in a finer detail, a measure called Hurst exponent is used. Fig. 5 shows the Hurst exponents of GEANT traffic. A Hurst exponent close to 0.5 indicates an uncorrelated series, and a Hurst exponent between 0.5 and 1 means long-term positive autocorrelation. Higher Hurst exponent values mean stronger correlation. Note that, GEANT traffic shows slightly stronger correlations than our synthetic traffic. In this work, we use the synthetic and random traffic. The temporal correlation in synthetic traffic is less than the traffic from GEANT topology.

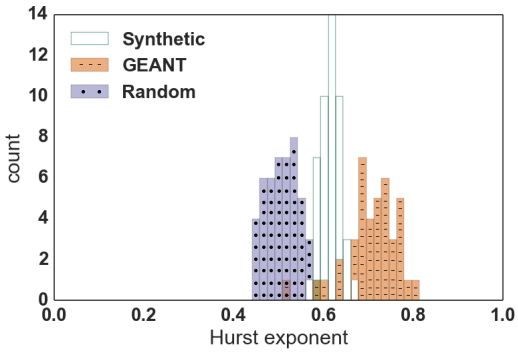


Fig. 5. The histogram shows the Hurst parameters for the real traffic trace from GEANT topology, and our synthetic traffic. An uncorrelated series has a Hurst exponent of 0.5 (the figure is from [1]).

B. Optimization Objectives

TE optimization metrics have been studied extensively. Altın et al. discusses that the link weight metric is the most important metric in the shortest path calculation [23]. In RFC 2328, the decision of choosing an appropriate metric has been left to the network operator. For example, Cisco routers assign inverse of the capacity of a link as the link weights.

In VTR, three common objectives are minimizing the maximum link utilization, minimizing the average weighted number of hops and minimizing average end-to-end delay [6]. Though many optimization objectives has been proposed, we limit our discussion to the most commonly used, and nonparametric ones. For example, Balon also analyzes a parametric objective called $Degrande$ [24], but finding the right parameters is another problem in itself. For that reason, in their conclusion they prefer $Delay$ over $Degrande$ even though those two objectives perform similarly.

B.1 Maximum Utilization (maxUtil)

Link utilization of a link can be described by $u_i = l_i/c_i$, where l_i is the load of the link i, and c_i is the capacity of the link. Then link utilization of the most heavily loaded link is denoted as $u_{\max}$.

Keeping $u_{\max}$ under 0.5 is a worthy effort, because such a topology can handle when the traffic doubles, without any congestion. However, the problem with $u_{\max}$ is its sensitivity to bottleneck links, as other people pointed out [4], [2]. Only one link value represents the overall topology can be problematic. Throughout this paper, we refer the objective of minimizing $u_{\max}$ as $maxUtil$.

B.2 Variance of Utilizations (varUtil)

The sensitivity of $u_{\max}$ can be solved by considering all link utilizations. Blanchy et al. proposed a metric that is directly related to the variance of link utilizations, which we refer as

```
1: procedure ADDNEWATTRACTOR( time t )
2:     Monitor current topology T_i
3:     Calculate performance metric α
4:     if α > α_max or ( α_old < α_th and α > α_th ) then        ▷ check the conditions to add a new attractor
5:         if α_old < α_th and α > α_th then        ▷ a transition has just been occurred from a bad state to a good state
6:             increment the pointer in the attractor array
7:         store T_i in A using storeTopology(A, T_)
```

Fig. 6. The modification we added to ASB algorithm. This improves the capability of finding attractors.

$varUtil$, as follows [25]

$$\text{varUtil} = \sum_{i \in E} (u_i - u_{\text{mean}})^2. \tag{5}$$

Here, u_{mean} is the average utilization of all links. The aim here is to balance the load across all links (i.e. E denotes the set of edges). This can be the best strategy when the traffic is uncorrelated temporally, or when the traffic follows a uniform distribution spatially. However, it has been shown that real network traffic follows a log-normal distribution spatially [26]. Also, this metric can work well under topology failures, if the probability of node failures is equally likely. Note that this metric is referred as *Blanchy* in [4].

B.3 Average Weighted Number of Hops (weightedHop)

Along with $maxUtil$, average weighted number of hops is most common metric in topology optimization. We simply refer it as *weightedHop*. $weightedHop$ is the average number of hops traversed by one unit traffic [6]. It is a traffic weighted hop count, rather than the pure hop count. Balon and his colleagues used minimum hop count in their work [4].

B.4 Delay

Elwalid et. al. discusses that a natural choice for the link cost is delay [27], and it can be calculated by:

$$Delay = \sum_{i \in E} \frac{1}{c_i - l_i}. \tag{6}$$

B.5 Normalized Available Bandwidth ($NABW$)

In traffic engineering context, a method called minimum interference routing algorithm (MIRA) has been introduced previously [3]. The authors propose an objective function to maximize available bandwidth on all possible pairs. In MIRA, the basic motivation is to maximize the probability of accepting future traffic demands. It is not possible to apply directly MIRA in topology optimization context because of the inherent differences of topology optimization with TE. However, we propose a new algorithm called normalized available bandwidth (NABW), which tries to achieve similar goal of handling of maximum future demands.

Let's assume that $l_{\max}(i,j)$ is the maximum utilization on the path $i-j$, we define an objective function for each path $i-j$ as

$$NABW(path_{i,j}) = \frac{1 - l_{\max}(i,j)}{\text{\# of paths passing through } l_{\max}(i,j)} \tag{7}$$

then, we sum for all paths as

$$total(NABW) = \sum_{\forall i,j \in N} NABW(path_{i,j}). \tag{8}$$

Here, $l_{\max}(i,j)$ corresponds to the load on the maximally loaded link between the path i and j. This link is basically the bottleneck link in m$path_{i,j}$. In short, we calculate available bandwidth normalized by the number of paths passing through the most heavily loaded links in that path. The intuition is, if there are more paths passing through a link, then that link has to have more importance than another link having the same amount of residual bandwidth. Our goal is to maximize $total(NABW)$.

C. Modifying ASB

Due to the inherent nature of ASB, we must make a few modifications to provide fairness for different objectives. In ASB, α is calculated by

$$\alpha = \frac{1}{1 + e^{50(u_{\max} - 0.5)}}. \tag{9}$$

For other metrics, we need to have a similar mapping to [0,1] range. However, for metrics like weightedHop, this mapping is not straightforward. Unlike $u_{\max}$, it is not possible to know what can be a good $weightedHop$ for a given traffic demand and topology. Theoretically the lower bound for weightedHop can be 1, but it is hard to find an upper bound and come up with a mapping function from weightedHop to α. This is also true for $Delay$, $NABW$ and $varUtil$.

First we need to determine whether a topology is "good" or "bad". To do that, we need to get a sense of what is an average performance for a given metric. First we run the simulator for sometime, a warm-up phase, and record the maximum and minimum values for a given objective function. Then, after warm-up phase ends, we start the simulation and normalize values using the recorded minimum and maximum values.

The approach we take on this work in detail is as follows. In the warm-up phase, the first 200 rounds, we record the minimum and maximum seen values such as $NABW_{\min}$ and $NABW_{\max}$. Then, after the warm-up period, we can calculate α as

$$\alpha = \frac{NABW - NABW_{\min}}{NABW_{\max} - NABW_{\min}}. \tag{10}$$

This way, we calculate a normalized α value for all optimization objectives. In addition, we also improved the attractor finding capabilities of ASB by making a small change as depicted in Fig. 6.

Table 1. Comparison of objectives.

objective	$u_{\max}$	total ABW	weighted hop
maxUtil	0.53	*0.50*	2.07
NABW	0.53	0.54	2.05
Delay	**0.33**	**0.71**	**1.94**
varUtil	*0.56*	0.54	*2.09*
weightedHop	0.49	0.68	2.01

Fig. 6 shows that the to add an attractor, the current value of α must be bigger than the $alpha_{\max}$, the highest value seen so far (line 4). However, instead of updating in a FIFO fashion as in original implementation of ASB, the suggested change in line 5 ensures that the new attractor added to the next slot. This modification prevents attractors being too similar, or too close to each other.

V. SIMULATION RESULTS

In this section, we first present the simulation settings, then present our results.

A. Settings

In the simulations, we used our own optical network simulator, which was developed by our group and written in C#. The simulator considers the resource availability in terms of ports and wavelengths, if resources are insufficient the path is not established. The simulations has been randomized using different seeds, number of ports and number of attractors. The physical topology consists of 100 nodes. Dijsktra's shortest path algorithm was used for both traffic and lightpath routing. We used two traffic model: synthetic and random. However, both models follow a log-normal distribution spatially, as it is the case observed in real network traces [26].

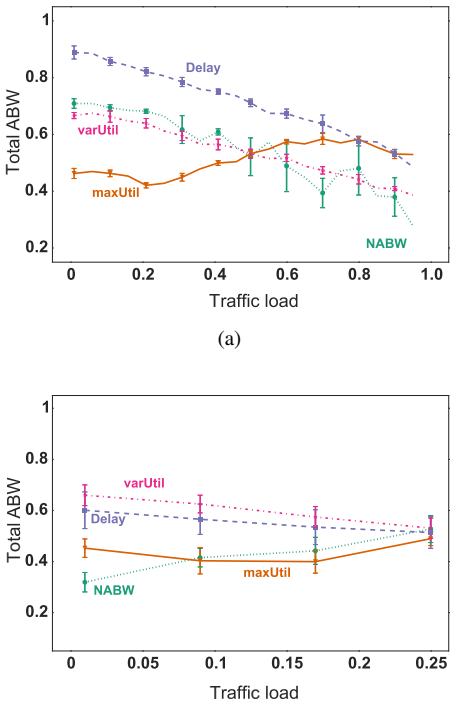
Cisco suggests operators to upgrade the network if the mean utilization of all the links exceeds 0.5 [28]. In analyzing the effect of traffic load, we increased traffic load to the point where mean utilization reached 0.5 since we are interested in typical operating regime.

B. Results

We first present results with synthetic traffic. Table 1 presents the results of simulations for three well established performance metrics. The best performances are shown in bold, while the worst performances in italic. $Delay$ performs best on all three metrics. $weightedHop$ performs very close to $Delay$ in total available bandwidth (ABW).

All these results confirm that $Delay$ is the best optimization metric. In that sense, our results agree with the previous work [4]. However, our results differ slightly. For example, we find that to minimize $u_{\max}$, the best metric is still $Delay$, instead of $maxUtil$. This result can be attributed to dynamic traffic. On the other hand, under static traffic, we expect similar results [4]. Since the previous work only focused on two traffic matrices, next we look how the traffic load affects the performance.

First, we look at the total ABW as the traffic load increases under synthetic traffic. Fig. 7(a) shows $Delay$ outperforms all other objectives, however, at high loads again $maxUtil$ comes closer to $Delay$.



(a)

(b)

Fig. 7. Total ABW under synthetic traffic , and random traffic. Under synthetic traffic, $Delay$ achieves much higher ABW than all the other metrics. Under random traffic, $varUtil$ performs best as expected. $weightedHop$ is not shown, as it performs very similar to $Delay$ in both scenarios. Also note that the x axis (traffic load) is narrower for random traffic. The bars correspond to 95% confidence intervals (the left figure is from [1]): (a) Synthetic traffic and (b) random traffic.

Fig. 7(b) shows the comparison under the random traffic model. In this model, we expect the performance of ASB to degrade, because the traffic is no longer correlated temporally. However, we still expect it to work, as the traffic is spatially distributed, as the spatial correlation still holds. From the figure, we notice that now the network reaches saturation for a load of 0.25. As one may expect, $varUtil$ performs best, and again $maxUtil$ performance is worse than others. Also, the total ABW is lower for all algorithms under random traffic with respect to static traffic.

Fig. 8 shows the maximum utilizations against traffic load. As the figure shows, $Delay$ performs best for minimizing maximum link utilization, and $maxUtil$ comes closer under very heavy traffic loads. This result also shows how our work differs from previous work [4]. That is, we use dynamic traffic, which changes every round. Then we try to optimize based on the traffic matrix of previous round, we do not necessarily expect the objective function to achieve the best result in its related metric. For example, $maxUtil$ does not achieve best $u_{\max}$, likewise $weightedHop$ does not achieve best weighted hop. Note that, NABW and $varUtil$ overloads the network as their maximum utilization exceeds 1 for heavy traffic loads. A utilization over 1 means overload, that is, the topology cannot accommodate the

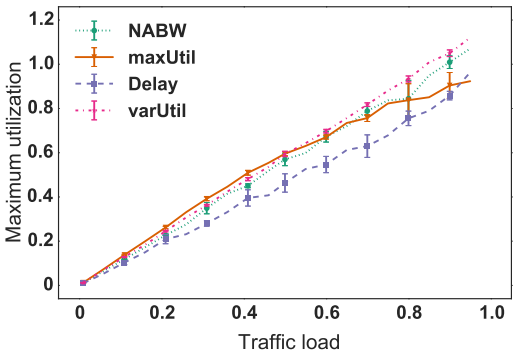


Fig. 8. Maximum utilization for different optimization objectives as the traffic load is varied. The bars correspond to 95% confidence intervals (figure is from [1]).

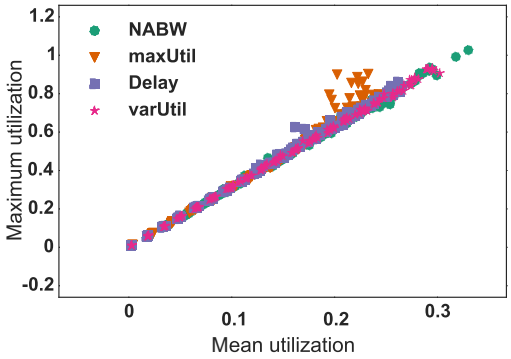


Fig. 9. Mean utilization vs. maximum utilization. $varUtil$ shows the least variance. A random subset of simulations is depicted for the sake of clarity.

traffic demand.

We were also interested how balanced topologies, each objective generates. Fig. 9 compares maximum utilization against mean utilization. Here, $varUtil$ is bounded in a narrow band, showing the least variation, as we expected. Conversely, $maxUtil$ is the metric with highest variation, confirming its high locality. $weightedHop$ is not depicted in the figures since it shows a very similar behavior to $Delay$.

Overall, $maxUtil$ performs poorly under random and correlated traffic models. On the other hand, $Delay$ performs best in both cases, for all three metrics we considered. $varUtil$ can also be a viable option in some cases such as under random traffic.

VI. RELATED WORK

There has been only a single work that evaluated various optimization objectives, which compared the various traffic engineering objective functions with static traffic matrices [4]. Using linear programming, the authors considered linear and nonlinear objective functions. For nonlinear objective functions, such as delay, they resorted to linear approximations.

The conclusion of Balon and his colleagues work guided some subsequent works to more informed studies [29]. The interaction between TE and congestion control has been analyzed, and an algorithm has been developed to improve stability [29].

An extensive survey by Wang et al. presents several algorithms on routing optimization for TE [30]. From the survey, the diversity of traffic optimization objectives can be seen. Though some of the objectives are not applicable to topology optimization, the work can constitute a starting point in understanding the diversity of objectives.

On topology optimization, some researchers used mixed-integer linear programming (MILP). However, there are some drawbacks of using linear programming. The formulation is NP-complete [31], and for large networks (i.e. more than 10 nodes) it becomes computationally intractable [6]. However, typical topologies are generally an order larger, for example AT&T topology consists of 154 nodes [32].

In addition to performance issues with MILP methods, a few metrics we propose here cannot be formulated as a linear programming problem since they are nonlinear, such as mean delay. Balon and his colleagues addressed this problem by using linear approximation [4], but they also highlighted the drawback of the linear approximation.

In another work, researchers were interested about which metrics matter more on video quality-of-experience (QoE) by using machine learning [33]. Since the Internet traffic will be dominated by video traffic in the near future, we believe the results we put here can lead to other studies on relating the QoE metrics to network-wide optimization metrics.

VII. CONCLUSION

Little attention has been paid to understanding the worthiness of different optimization objectives. Only in one work, using linear programming some researchers evaluated the efficiency of such objectives. However, the use of linear approximation of nonlinear objective functions can be problematic. In addition, even though it was suggested that $Delay$ is a better optimization objective than $maxUtil$, the research community has been sticking with $maxUtil$.

We compared different topology optimization metrics using machine learning. Comparison of optimization objectives and use of machine learning are two novel aspects of this study. Use of machine learning is especially crucial, as it strives to provide a fair framework for all objective functions. We found out that $Delay$ is the best metric, which is in agreement with the conclusions of Balon and his colleagues. However, our conclusion is more comprehensive. Instead of static traffic, we used a dynamic traffic and analyzed the 33 days of traffic. In the end, we showed the predictive capability of different objective functions. Even though we took a different approach, our conclusion agrees with the previous work.

The second most consequential finding in this paper is the sensitiveness of maximum utilization. Under low traffic loads, there are more worthy objective functions than minimizing maximum utilization. Though, the rule of thumb of keeping maximum link utilization under 0.5 is an intuitive proactive bandwidth allocation strategy, there are better objective functions that can achieve the same goal. $Delay$ is one of them, the metric $NABW$ we propose here is also quite reasonable for low loads. $varUtil$ looks like the best solution for random traffic.

The simulation results confirm the concerns about the locality of maximum utilization using realistic traffic traces.

REFERENCES

[1] Y. S. Hanay, S. Arakawa, and M. Murata, "Evaluation of topology optimization objectives," in *Proc. IEEE LCN*, Clearwater, 2015, pp. 458–461.

[2] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 4, pp. 756–767, 2002.

[3] K. Kar, M. Kodialam, and T. V. Lakshman, "Minimum interference routing of bandwidth guaranteed tunnels with MPLS traffic engineering applications," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 12, pp. 2566–2579, 2000.

[4] S. Balon, F. Skivee, and G. Leduc, "How well do traffic engineering objective functions meet TE requirements?," in *Proc. IFIP*, vol. 3976, May 2006, pp. 75–86.

[5] A. K. Dutta, N. K. Dutta, and M. Fujiwara, *WDM technologies : Optical networks. Volume III*. Elsevier Academic Press, 2004.

[6] J. Zheng and H. T. Mouftah, *Optical WDM Networks: Concepts and design principles*. Piscataway, NJ: John Wiley & Sons, 2004.

[7] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized? state distribution trade-offs in software defined networks," in *Proc. HotSDN*, 2012, pp. 1.

[8] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2211–2219.

[9] A. Sivaraman, K. Winstein, P. Thaker, and H. Balakrishnan, "An experimental study of the learnability of congestion control," in *Proc. ACM SIGCOMM*, Aug. 2014, pp. 479–490.

[10] Y. Jin, N. Duffield, J. Erman, P. Haffner, S. Sen, and Z.-L. Zhang, "A Modular machine learning system for flow-level traffic classification in large networks," *ACM Trans. Knowl. Discovery from Data*, vol. 6, no. 1, pp. 1–34, 2012.

[11] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches," *Computer Commun.*, vol. 34, no. 18, pp. 2227–2235, 2011.

[12] W. E. Leland, M. S. Taqqu, and D. V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Trans. Netw.*, vol. 2, no. 1, pp. 1–15, 1994.

[13] T. Karagiannis, M. Molle, M. Faloutsos, and a. Broido, "A nonstationary Poisson view of Internet traffic," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 1558–1569.

[14] P. Borgnat, G. Dewaele, K. Fukuda, P. Abry, and K. Cho, "Seven years and one day: Sketching the evolution of internet traffic," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 711–719.

[15] W. B. Gong, Y. Liu, V. Misra, and D. Towsley, "Self-similarity and long range dependence on the internet: A second look at the evidence, origins and implications," *Comput. Netw.*, vol. 48, no. 3, pp. 377–399, 2005.

[16] Y. Koizumi, T. Miyamura, S. Arakawa, E. Oki, K. Shiomoto, and M. Murata, "Adaptive virtual network topology control based on attractor selection," *IEEE/OSA J. Lightw. Technol.*, vol. 28, no. 11, pp. 1720–1731, 2010.

[17] R. Rojas, *Neural networks: A systematic introduction*. New York: Springer-Verlag, 1996.

[18] D. O. Hebb, *The organization of behaviour*, New York: Wiley, 1949.

[19] N. N. Aizenberg and I. N. Aizenberg, "CNN based on multi-valued neuron as a model of associative memory for grey scale images," in *Proc. IEEE CNNA workshop*, 1992, pp. 36–41.

[20] G. Tanaka and K. Aihara, "Complex-valued multistate associative memory with nonlinear multilevel functions for gray-level image reconstruction," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1463–1473, 2009.

[21] D. O. Awduche and J. Agogbua, "RFC 2702: Requirements for traffic engineering over MPLS," tech. rep.

[22] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, p. 83, 2006.

[23] A. Altin, B. Fortz, M. Thorup, and H. Ümit, "Intra-domain traffic engineering with shortest path routing protocols," *Annals of Operations Research*, vol. 204, no. 1, 2013.

[24] N. Degrande, G. Van Hoey, P. de La Vallée Poussin, and S. Van den Bosch, "Inter-area traffic engineering in a differentiated services network," *J. Netw. Sys. Manag.*, vol. 11, pp. 427–445, 12 2003.

[25] F. Blanchy, L. Mélon, and G. Leduc, "An efficient decentralized on-line traffic engineering algorithm for MPLS networks," in *Proc. ITC*, Berlin, Germany, 2003.

[26] A. Nucci, A. Sridharan, and N. Taft, "The problem of synthetically generating IP traffic matrices: initial recommendations," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 19–32, 2005.

[27] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: Multipath adaptive traffic engineering," *Comput. Netw.*, vol. 40, no. 6, pp. 695–709, 2002.

[28] CISCO, "Best practices in core network capacity planning architectural principles of the MATE portfolio of products," tech. rep., 2013.

[29] J. He, J. Rexford, and M. Chiang, "Don't optimize existing protocols, design optimizable protocols," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 3, pp. 53–58, 2007.

[30] N. Wang, K. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for internet traffic engineering," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 1, pp. 36–56, 2008.

[31] I. Chlamtac, A. Ganz, and G. Karmi, "Lightpath communications: An approach to high bandwidth optical WAN's," *IEEE Trans. Commun.*, vol. 40, no. 7, pp. 1171–1182, 7 1992.

[32] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz, "On realistic network topologies for simulation," in *Proc. ACM SIGCOMM workshop* , Karlsruhe, Germany, 2003, pp. 28–32.

[33] A. Balachandran, V. Sekar, and A. Akella, "A quest for an internet video quality-of-experience metric," in *Proc. ACM Workshop*, Redmond, Washington, 2012, pp. 97–102.

Y. Sinan Hanay received his Ph.D. in Electrical and Computer Engineering from University of Massachusetts, Amherst in 2011. He worked at NICT Japan and Osaka University between 2011 and 2016, and in TED University between 2016 and 2017. Currently, he is an Assistant Professor at Erzurum Technical University, Turkey. He is a co-author of a paper which received the best paper award in IEEE HPSR conference in 2011. His research interests include network systems design and machine learning.

Shin'ichi Arakawa received M.E. and D.E. degrees in Informatics and Mathematical Science from Osaka University in 2000 and 2003. From August 2000 to March 2006, he was an Assistant Professor with the Graduate School of Economics, Osaka University, Japan. In April 2006, he moved to the Graduate School of Information Science and Technology, Osaka University, Japan. He has been an Associate Professor from October 2011. His research interests include optical networks and complex networks. He is a member of IEEE and IEICE.

Masayuki Murata received his M.E. and D.E. in Information and Computer Sciences from Osaka University in 1984 and 1988. In April 1984, he joined the Tokyo Research Laboratory of IBM Japan as a Researcher. From September 1987 to January 1989, he was an Assistant Professor with the Computation Center of Osaka University. In February 1989, he moved to the Department of Information and Computer Sciences of the Faculty of Engineering Science of Osaka University. From 1992 to 1999, he was an Associate Professor with the Graduate School of Engineering Science at Osaka University, and since April 1999, he has been a full Professor. He moved to the Graduate School of Information Science and Technology of Osaka University in April 2004. He has had more than 300 papers published in international and domestic journals and conferences. His research interests include computer communication networks, performance modeling, and evaluation. He is a member of IEEE, the Association for Computing Machinery (ACM), The Internet Society, and IPSJ.