Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Enhancement of Extreme Learning Machine for Estimating Blocking Probability of OCS Networks with Fixed-Alternate Routing

SHUO LI[1], (Member, IEEE), HO CHUN LEUNG[2], ERIC W. M. WONG[2],(SENIOR MEMBER, IEEE), and CHI SING LEUNG[2],(SENIOR MEMBER, IEEE)

[1]S. Li is with the School of Engineering, RMIT University, Melbourne, Australia (email: shuo.li2@rmit.edu.au).

[2]H.C. Leung, Eric W.M. Wong and C.S. Leung are with the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong (email: hcleung35-c@my.cityu.edu.hk, eeewong@cityu.edu.hk and eeleungc@cityu.edu.hk).

Corresponding author: C.S. Leung (e-mail: eeleungc@cityu.edu.hk).

The work was mainly supported by a research grant (CityU 11259516) from the Hong Kong Special Administrative Region and was partially supported by two research grants from City University of Hong Kong (Project No.:7004829 and 7004842).

ABSTRACT

In a previous work, we proposed a neural network approach to estimate the blocking probability of optical networks with fixed routing. The neural network was implemented by the extreme learning machine (ELM) framework, in which the training inputs were the optical network parameters, and the output was the overall blocking probability. Numerical results showed that the neural-network-based estimation was accurate and thousands of times faster than computer simulation. In this paper, we apply the neural network approach to optical circuit switching (OCS) networks with fixed-alternate routing and improve the training method by using an enhancement of ELM framework. Unlike the previous ELM framework, the enhancement of ELM framework provides a random-search based selection phase for the hidden nodes during the training step. As a result, similar performance can be achieved using fewer hidden nodes than the previous ELM framework. Numerical results show that the new enhancement of ELM training algorithm provides more accurate blocking probability estimates while reducing the required number of hidden nodes by a third compared to the previous ELM training algorithm. Furthermore, for some light traffic loading situations, our new training algorithm is hundreds of times more accurate than the existing well-know analytical approximation method.

INDEX TERMS Artificial Neural Network; Blocking Probability; Optical Circuit Switching Network; Fixed-Alternate Routing.

I. INTRODUCTION

Circuit switching is widely used in telephony, and has played an important role in optical networks. For an optical circuit switching (OCS) network, an end-to-end lightpath must be established before transmitting the data; otherwise, the connection request will be blocked. To reduce the occurrence of blocking, various approaches for alternate routing were developed [1]–[3]. For instance, fixed-alternate routing proposed in [1] allowed a connection request that was blocked on its desired route to overflow to an alternate route according to the predefined order of a list of alternate routes. In fact, evaluating the blocking probability helps network engineers to conduct load balancing and congestion control [4]. A common way of estimating the blocking probability is by computer simulations [2], [5], [6]. However, simulations are time-consuming, especially for large optical networks. Another evaluation approach is analytical approximation, such as the well-known Erlang Fixed Point Approximation (EFPA) [7]. The EFPA can estimate the blocking probability in an efficient manner due to its simple model structure but the simplifying assumptions result in reduced accuracy of the approximation.

On the other hand, we propose to use a neural network for learning the mapping from the parameters of an optical

network to the blocking probability. This method allows us to estimate the blocking probability in a second, which is thousand times faster than that of computer simulations. With this advantage, this method can be applied in the network design phase and in a dynamic scenario. In the dynamic scenario, the network needs to change the routing table in real time, and several routing tables may be provided by different routing and wavelength assignment algorithms. Our method rapidly estimates the blocking probability for all options, and recommends the best one with the lowest blocking probability to the network management system.

In this paper, we propose to use the enhancement of ELM framework instead of the ELM framework. Under this framework, where the hidden nodes were randomly generated, the universal approximation ability was demonstrated in [8]–[10]. The enhancement of error minimized ELM (EEM-ELM) algorithm [11] is used to train a single-hidden-layer feedforward network (SLFN) model with the training objective of minimizing the mean square error (MSE) of the logarithm of the blocking probability. Indeed, the incremental ELM (I-ELM) algorithm used in our previous paper [12], which adds the hidden nodes without selection, results in a large neural network with many redundant hidden nodes. This decreases the efficiency and accuracy of the neural network.

However, the EEM-ELM algorithm does not contain the above drawbacks due to a random-search based selection process in each incremental stage for the addition of new hidden nodes. This approach allows us to select the hidden nodes that most significantly reduces the estimation error, and hence reduces the required number of hidden nodes while increasing the accuracy of the estimation. After that, the alternating direction method of multipliers (ADMM) [13], [14] is used to further remove the remaining unimportant nodes. The contributions of this paper are summarized as follows.

- For the first time, a neural network approach is applied to estimate blocking probability in an OCS network with fixed-alternate routing.
- The EEM-ELM is adopted to improve the accuracy of the estimation and reduce the required number of hidden nodes.
- Numerical results demonstrate that our training algorithm provides accurate blocking probability estimates while reducing the required number of hidden nodes by a third compared to the algorithm in our previous paper.
- For a lightly-loaded traffic network with fixed-alternate routing, the estimated blocking probability from our algorithm can be hundreds of times more accurate than that of EFPA.

The remainder of the paper is organized as follows. Section II briefly reviews the related work. Section III introduces our proposed training model. Section IV describes the simulation setup. Section V describes the ELM training process and the results from the experiments. Finally, conclusions are given in Section VI.

II. RELATED WORK

A. OCS NETWORKS WITH FIXED-ALTERNATE ROUTING

In an OCS network, a lightpath between the source destination (S-D) pair is required to be established for a connection before transmitting the data. To establish a lightpath, a wavelength needs to be reserved on each of the intermediate links along the path. For each connection request, the source node sends a control packet through the primary path to reserve the wavelength. When the wavelength is successfully reserved on all links in the lightpath, the lightpath is established and the data can be transmitted, otherwise, the connection is blocked by the primary path.

If fixed-alternate routing is used, the OCS network is considered as a layered structure, where the Layer 1 (primary layer) contains the primary paths of all S-D pairs, Layer 2 (the first overflow layer) contains the secondary paths of all S-D pairs, and so on. If the connection request is blocked by Layer 1, the source node tries to overflow the connection request to Layer 2, i.e. by establishing a lightpath through the secondary path [5], [15]. This process repeats until the lightpath is established, or the number of allowable paths is reached. In the latter case, since no available lightpath can be established, the connection request is blocked and cleared from the OCS network. For simplicity, we assume that the network is purely all-optical, such that the same wavelength is used along the entire end-to-end path.

B. NETWORKING APPLICATION OF NEURAL NETWORKS

Nowadays, many neural network approaches are applied to solve networking problems. For example, an neural network approach was proposed for the optical fiber channel modelling in an optical network by embedding a deep fully-connected feed-forward neural network [16]. A significant improvement was observed in the intensity modulation direct detection (IM-DD) system. Merayo *et al*. [17] proposed a neural network approach to adjust the proportional-integral-derivative (PID) controller for the bandwidth allocation in a passive optical network. Mo *et al*. [18] implemented a deep-neural-network-based method to estimate the power dynamics of a reconfigurable optical add-drop multiplexer (ROADM) system, and assign the wavelength for switching with least power consumption in an optical network.

Moreover, neural network can also be applied to the blocking probability estimation. Araujo *et al*. [19] introduced a multilayer perceptron (MLP) model, a classic neural network approach, to estimate the blocking probability of an OCS network. In our previous paper [12], we proposed to use the ELM framework to estimate the blocking probability of the optical burst switching (OBS) or optical packet switching (OPS) network with fixed routing, using the logarithm of the blocking probability for better performance of the neural network. We trained the neural network with our ADMM-Log-I-ELM algorithm, and the estimation results were shown to be

accurate. However, the algorithm inherits a drawback from the I-ELM algorithm, which adds the hidden nodes without selection. This results in a large number of hidden nodes required to conduct the estimation and hence the efficiency of the algorithm might be affected. To tackle this issue, this paper proposes to use EEM-ELM instead of I-ELM.

C. ELM AND ITS ENHANCEMENTS

In the ELM framework, we consider an SLFN model for the function approximation problem. The training set is denoted as $\mathbf{D}_t = \{(x_l; p_l) : x_l \in \mathbf{R}^d; p_l \in \mathbf{R}; l = 1; \ldots; N\}$, where x_l and p_l are the input vector and target output of the l-th sample, respectively. In [20]–[22], the network output of the SLFN approach is expressed as

$$f_n(x) = \sum_{i=1}^{n} \beta_i g_i(x) \tag{1}$$

where $g_i(x)$ is the output of the ith hidden node, and β_i is the weight between the ith hidden node and the output node.

For the hidden nodes in the ELM, the activation functions are defined as sigmoid functions [23]–[25]. In other words, the output of the ith hidden node is expressed as

$$g_i(x) = \frac{1}{1 + \exp\{-(a_i^{\mathsf{T}} x + b_i)\}}; \tag{2}$$

where a_i is the input weight vector of the ith hidden node, and b_i is the bias term of the ith hidden node. For the bias terms b_i's and the input weights a_i's, they are generated randomly with the concept of ELM [8], [9]. When n hidden nodes are used for approximation, the MSE is expressed as

$$E = \sum_{l=1}^{N} (p_l - \sum_{i=1}^{n} \beta_i g_i(x_l))^2 = \left\| p - \sum_{i=1}^{n} \beta_i g_i \right\|_2^2; \tag{3}$$

where $p = [p_1; \ldots; p_N]^{\mathsf{T}}$, and $g_i = [g_i(x_1); \ldots; g_i(x_N)]^{\mathsf{T}}$.

In [26], Feng et al. proposed an error minimized ELM (EM-ELM) algorithm. The idea of EM-ELM algorithm is an iterative process in which a certain number of hidden nodes is added in each iteration. For each iteration, when the new hidden nodes are inserted, the output weights are updated recursively without retraining the entire network. This reduces the complexity and running time of the training process. The training objective is expressed as follows:

$$E_n = \|p - \Omega_n \beta_n\|_2^2; \tag{4}$$

$$\Omega_n = \begin{bmatrix} g_1(x_1) & \cdots & g_n(x_1) \\ g_1(x_2) & \ddots & g_n(x_2) \\ \vdots & \vdots & \ddots \\ g_1(x_N) & & g_n(x_N) \end{bmatrix} \tag{5}$$

The hidden layer matrix Ω_{n+1} can be further expressed as $\Omega_{n+1} = [\Omega_n; \ldots; \ldots]$.

At the nth incremental learning step, a group of new nodes $\ldots_n$ is generated randomly, and added into the network. For the newly added nodes, we need to update the output weights β_n. The calculation of output weights is formulated as follows:

$$Q_n = ((I - \ldots_n \ldots_n^{y}) \Omega_n)^{y} \tag{6}$$

$$T_n = \ldots_n^{y} (I - \ldots_n Q_n) \tag{7}$$

$$\beta_{n+1} = \begin{bmatrix} T_n \\ Q_n \end{bmatrix} p \tag{8}$$

where y denotes the Moore-Penrose inverse.

Hereafter, Lan et at. [11] introduced the Enhancement of EM-ELM (EEM-ELM). The EEM-ELM modifies the EM-ELM by adding a random-search based selection phase. For the EEM-ELM algorithm, at each incremental learning step, j candidate sets of hidden nodes are randomly generated, and the output weights and the approximation error are calculated using the EM-ELM algorithm. From the j candidate sets of hidden nodes, the set yielding the smallest approximation error is added to the neural network.

D. ADMM

In [13], [14], the authors proposed the ADMM framework to solve the convex optimization problem using an iterative approach. The constraint optimization problem is shown as follows:

$$\begin{aligned} &\min_{\ldots;u} : \quad (\ldots) + (u) \\ &s.t: \quad C \ldots + D u = v; \end{aligned} \tag{9}$$

where and are the cost terms in the objective function, and and u are decision variable vectors in the optimization.

In the ADMM, an augmented Lagrangian is constructed, given by

$$\begin{aligned} L(\ldots; u; \ldots) = {} & (\ldots) + (u) + \ldots^{\mathsf{T}}(C \ldots + D u \quad v) \\ & + \frac{}{2} \| C \ldots + D u \quad v \|_2^2; \end{aligned} \tag{10}$$

where is a positive penalty parameter, and is the Lagrangian vector. Hence, the iterative process for $\{\ldots; u; \ldots\}$ is given by

$$\ldots^{k+1} = \arg\min_{\ldots} L(\ldots; u^k; \ldots^k); \tag{11a}$$

$$u^{k+1} = \arg\min_{u} L(\ldots^{k+1}; u; \ldots^k); \tag{11b}$$

$$\ldots^{k+1} = \ldots^k + \quad C \ldots^{k+1} + D u^{k+1} \quad v \quad : \tag{11c}$$

E. EFPA FOR OCS NETWORKS

In addition to simulation and neural network approaches, one traditional way of blocking probability evaluation in OCS networks is analytical approximation, e.g. EFPA. EFPA was originally developed for the blocking probability estimation in circuit-switched telephone networks by Cooper and Katz [27]. In [5], Wang et al. used EFPA to estimate the blocking probability in an OCS network with fixed-alternate routing.

The concept of EFPA is to decompose the network into independent links, and to assume that the arrival traffic to each link follows a Poisson process. This results in a fixed

point iterative solution for obtaining the blocking probability of each link. With this simple structure, EFPA can estimate the blocking probability of the network in a very fast manner. However, it was observed in [28], [29] that EFPA might introduce significant errors due to the assumptions of independent links and a Poisson arrival process to each link. Moreover, the convergence and uniqueness of solutions of EFPA are not always guaranteed [7], [30]. Hence, EFPA may not provide an accurate estimation of blocking probability for OCS networks with fixed-alternate routing.

III. OUR PROPOSED TRAINING MODEL

A. METHODOLOGY

In general, for an OCS network topology with some network parameters $\{x_l : x_l \in \mathbb{R}^d\}$, the corresponding blocking probability $\{p_l : p_l \in \mathbb{R}_{>0}\}$ can be obtained via computer simulation. In this paper, we use these d network parameters as input to represent the OCS network conditions and train an SLFN to learn the mapping from the network conditions to the target output, i.e. the blocking probability.

However, for an OCS network with different network parameters, the range of blocking probability can be large, e.g. $[10^{-5}; 10^{-1}]$, depending on network load, network topology, routing strategy and wavelength-assignment strategy. This may lead to large approximation errors for small blocking probability values if the unscaled blocking probability values (as generated by simulation) are used as the target output of the SLFN directly. The results of the ADMM-I-ELM algorithm in our previous paper [12] illustrates this problem. Therefore, in [12], we proposed to train our SLFN using the logarithm of the blocking probability, $\{\beta_l = \log p_l\}$, instead of p. The new training set is denoted as:

$$\mathbb{D}_t = \{(x_l; \beta_l = \log p_l) : x_l \in \mathbb{R}^d; \beta_l \in \mathbb{R}_{<0}; l = 1, \ldots, N\} \quad (12)$$

The training objective becomes

$$E = \|\boldsymbol{\beta} - \mathbf{y}\|_2^2; \quad (13)$$

where $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_N]^T$.

Based on the EEM-ELM, we develop a new ELM algorithm, namely Log-EEM-ELM. For minimizing the objective function E, the Log-EEM-ELM continuously inserts the hidden nodes into the SLFN until the target error is achieved. Algorithm 1 summarizes the steps of our proposed algorithm.

The Log-EEM-ELM algorithm eventually gives us n_e hidden nodes i.e. $\{g_1(x), \ldots, g_{n_e}(x)\}$. As the Log-EEM-ELM algorithm contains a selection phase for adding hidden nodes, this can guarantee that the selected nodes have a good error reduction ability. Nevertheless, some hidden nodes may be relatively unimportant. The ADMM approach is used to remove these unimportant nodes. Incorporating ADMM into our framework produces a new algorithm, namely ADMM-Log-EEM-ELM. The ADMM-Log-EEM-ELM algorithm first trains the SLFN using the Log-EEM-ELM and then uses the ADMM framework to remove the unimportant nodes. When the n_e hidden nodes are considered, the approximation error is denoted as

$$E = \|\boldsymbol{\beta} - \sum_{i=1}^{n_e} \beta_i \mathbf{g}_i\|_2^2; \quad (14)$$

where $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_N]^T$.

To remove the unimportant nodes, an ℓ_1 norm penalty term is required in the objective function. The modified objective function is denoted as

$$J = \frac{1}{2}\|\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1; \quad (15)$$

where λ is a tradeoff parameter between the number of selected nodes and the approximation error of the log blocking probability. λ can be tuned to trade off the number of hidden nodes with the approximation accuracy. When λ is large, the SLFN contains fewer hidden nodes. When λ is small, the SLFN has greater approximation accuracy. Due to the presence of the penalty term, the output weights for the unimportant nodes are reduced to zero. In other words, the influence of these nodes disappears and they can be removed afterwards.

The ADMM framework can be used to solve the unconstrained optimization in (15). A dummy variable $\boldsymbol{u}$ is introduced, such that the optimization problem becomes a constrained one. The formulation of the problem becomes

$$\begin{aligned} &\min_{\boldsymbol{\beta}, \mathbf{u}} \ \tfrac{1}{2}\|\boldsymbol{\beta} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{u}\|_1; \\ &s.t.: \ \boldsymbol{\beta} = \mathbf{u}: \end{aligned} \quad (16)$$

Algorithm 1 Log-EEM-ELM

1: Initialization: Let L_0 denotes the initial number of hidden nodes.
2: Define the termination condition ε.
3: Compute the initial hidden layer output matrix $\mathbf{H}_0$.
4: Compute the initial output weights $\boldsymbol{\beta}_0$: $\boldsymbol{\beta}_0 = \mathbf{H}_0^{\dagger}\mathbf{y}$, where $\mathbf{y} = [\beta_1, \ldots, \beta_N]^T$.
5: Let $n = 0$, and compute the approximation error ε_n.
6: **while** $L_n < L_{max}$ and $\varepsilon < \varepsilon_n$ **do**
7: **for** i=1:j **do**
8: Construct a set of L_n hidden nodes to the network, so that the total number of hidden nodes becomes $L_{n+1} = L_n + L_n$. Hence, the hidden layer matrix becomes $\mathbf{H}_{n+1}^i = [\mathbf{H}_n, \mathbf{H}_n^i]$
9: Compute the new output weights as follows:
$\mathbf{Q}_n^i = ((\mathbf{I} - \mathbf{H}_n \mathbf{H}_n^{\dagger})\mathbf{H}_n^i)^{\dagger}$
$\mathbf{T}_n^i = \mathbf{H}_n^{\dagger}(\mathbf{I} - \mathbf{H}_n^i \mathbf{Q}_n^i)$
$\mathbf{H}_{n+1}^i = \begin{bmatrix} \mathbf{T}_n^i \\ \mathbf{Q}_n^i \end{bmatrix}$ $\boldsymbol{\beta}$
10: Compute the approximation error ε_{n+1}^i.
11: **end for**
12: $n = n + 1$.
13: Select a set of hidden nodes j yielding the smallest approximation error, and $\mathbf{H}_n = \mathbf{H}_n^j$, $\boldsymbol{\beta}_n = \boldsymbol{\beta}_n^j$, and $\varepsilon_n = \varepsilon_n^j$
14: **end while**

The augmented Lagrangian of (16) is denoted as

$$L(; u;) = \frac{1}{2} k\beta \quad k_2^2 + \quad ku k_1 + \quad ^{\mathsf{T}}(u \quad) + \frac{}{2} ku \quad k_2^2: \quad (17)$$

Hence, the corresponding ADMM update of , u and are respectively given by (19), (21) and (23), as derived below.

Computing $^{k+1}$:

By fixing u as u^k and as ^{k} in (17), $^{k+1}$ can be updated by

$$\begin{aligned} ^{k+1} &= \arg\min_{w} L(; u^k; ^k) \\ &= \arg\min \frac{1}{2} k\beta \quad k_2^2 + \quad ^{k\mathsf{T}}(u^k \quad) \\ &\quad + \frac{}{2} ku^k \quad k_2^2 \\ &= \arg\min \frac{1}{2} k\beta \quad k_2^2 + \frac{}{2} ku^k \quad + \frac{1}{} \quad ^k k_2^2 \end{aligned} \quad (18)$$

From (18), we obtain

$$^{k+1} = \left[\quad ^{\mathsf{T}} \quad + I \right]^{-1} \left[\quad ^{\mathsf{T}} \beta + u^k + \quad ^k \right]; \quad (19)$$

where I is an identity matrix.

Computing u^{k+1}:

By fixing as $^{k+1}$ and as ^{k} in (17), u^{k+1} can be updated by

$$u^{k+1} = \arg\min_{u} \quad ku k_1 + \frac{}{2} ku \quad ^{k+1} + \frac{1}{} \quad ^k k_2^2: \quad (20)$$

The term $ku k_1$ in (20) is non-differentiable; however, the closed-form solution can be obtained by using sub-differential calculus [14]. The formulation is denoted as

$$u^{k+1} = S_{=} \quad ^{k+1} \quad \frac{1}{} \quad ^k \quad : \quad (21)$$

where S ( ) is a soft-threshold operator [14], given by

$$S( ) = \mathrm{sgn}( ) \quad (j \; j \quad): \quad (22)$$

Note that the soft-threshold operator is an elementwise operator.

Computing $^{k+1}$:

$$^{k+1} = \quad ^k + \quad u^{k+1} \quad ^{k+1} \quad : \quad (23)$$

In short, we notice that the computational complexity of ADMM-Log-EEM-ELM for each iteration is $O(n^2 N) + O(n^2)$.

B. SIMULATION AND TRAINING PARAMETERS

For the neural network training inputs, we consider seven parameters for an OCS network, as defined in Table I. They are similar to our previous paper [12], except for the addition of P, and the renaming of C as F. The seven parameters are all independent parameters and represent the properties of the network input traffic load (i.e. E and D), the optical links (i.e. W and F), and the routing scheme (i.e. P, APL and CR) respectively. As the fixed-alternate routing is considered in this paper, the parameters APL and CR are transformed into vector form, i.e. APL and CR. The detailed explanation for the seven parameters is as follows.

First, E denotes the mean traffic load per S-D pair, measured in Erlangs, and is the most important factor to to determine the blocking probability, as blocking increases with E. Second, D denotes the difference between the maximum and the minimum offered load to each S-D pair. We assume the offered load to each S-D pair is uniformly distributed within the interval $[E \quad D/2; E + D/2]$. Note that increasing D with E fixed, the blocking probability of the network increases.

Third and fourth, F and W denote the number of fiber pairs on each link and the number of wavelengths per fiber, respectively. Each fiber has the same set of W wavelengths; hence, the number of wavelength in each fiber is W. Note that increasing W with other parameters fixed, the blocking probability of the network decreases, as does increasing F with other parameters fixed.

Fifth and sixth, APL and CR are vectors denoting the average path length and concentration of routes, respectively. As we consider OCS networks with one primary layer and two overflow layers, the two vectors become $\{APL_1; APL_2; APL_3\}$ and $\{CR_1; CR_2; CR_3\}$. The value APL_i denotes the average path length in hops for Layer i paths between the S-D pairs, which is computed based on the routing table. Note that increasing APL_i typically increases the blocking probability [31]. The value CR_i, as defined in Araujo *et al.* [19], relates to the maximum distance among a set of nonzero values, which are the number of Layer i paths passing through a link (zero values are excluded from the set). It reflects the load balancing of the traffic in the network, which helps to reduce the blocking probability. Therefore, a lower CR_i value corresponds to better load balancing, and results in lower blocking probability.

Seventh, P represents the number of allowable paths. P is related to the number of the allowable overflows. For a given

TABLE 1: Independent Input Parameters

Parameter	Definition
E	Mean traffic load per S-D pair (Erlang)
D	Difference between max and min traffic load per S-D pair
W	No. of wavelengths per fiber
F	No. of fiber pairs per link
APL : $\{APL_1; APL_2; APL_3\}$	Average path length in the three layers
CR : $\{CR_1; CR_2; CR_3\}$	Concentration of route in the three layers
P	No. of allowable paths

TABLE 2: Trained models with various parameters

Input Parameters	MSE
E, D, W, F, fCR_1g, $fAPL_1g$	1:491 10^{-3}
E, D, W, F, P, fCR_1g, $fAPL_1g$	6:743 10^{-4}
E, D, W, F, P, $fCR_1; CR_2g$, $fAPL_1; APL_2g$	5:879 10^{-4}
E, D, W, F, P, $fCR_1; CR_2; CR_3g$, $fAPL_1; APL_2; APL_3g$	4:664 10^{-4}

P, a network allows a connection request to overflow $P - 1$ times. In fact, an S-D pair can use $min(P; Rm)$ paths, where Rm denotes the maximum number of available paths for S-D pair m. For a network with light traffic load, increasing P value provides more chances for a request to establish a lightpath, thus reducing the blocking probability. However, if the traffic load E is high and $APL_3 > APL_2 > APL_1$, increasing P might force the overflowed connection requests to use longer paths, hence the blocking probability would be increased.

In our previous work [12], the significance of E, D, W, F (the same meaning as C in [12]), APL and CR were justified by the MSE changes with respect to the changes in number of parameters. All six parameters were significant to our training model. In this paper, which considers the fixed-alternate routing, the number of allowable paths P is introduced, and APL and CR are transformed into vector form $fAPL_1; APL_2; APL_3g$ and $fCR_1; CR_2; CR_3g$.

The significance of these new input parameters is shown in Table 2. It shows that by including P, the MSE has a significant reduction, which decreases from $1{:}491 \quad 10^{-3}$ to $6{:}743 \quad 10^{-4}$. With the additional layer information for APL_2 and CR_2, the MSE further reduces to $5{:}879 \quad 10^{-4}$. With additional information for APL_3 and CR_3, the MSE further reduces to $4{:}664 \quad 10^{-4}$. These results demonstrate that these new input parameters are essential for our training model to provide an accurate estimation of blocking probability.

IV. SIMULATION SET-UP AND NEURAL NETWORK TRAINING

A. OCS NETWORK SIMULATIONS

For the simulation, we implemented a Markov-chain simulation model. In the simulation, the arrival process and the request duration are assumed to be Poisson and exponentially distributed, respectively. Also, we assume all paths are all-optical end-to-end, which means no wavelength regeneration. Therefore, the wavelength continuity must be maintained [32], [33], i.e. the same wavelength is used for all links along the path. This assumption may not be realistic in a network of large geographic extent, but it allows us to simplify the problem. When establishing a lightpath, the wavelength is chosen randomly from the set of available wavelengths.

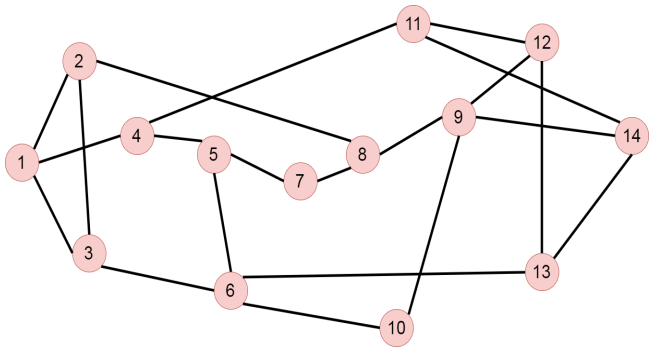

FIGURE 1: 14-node NSFNET Topology

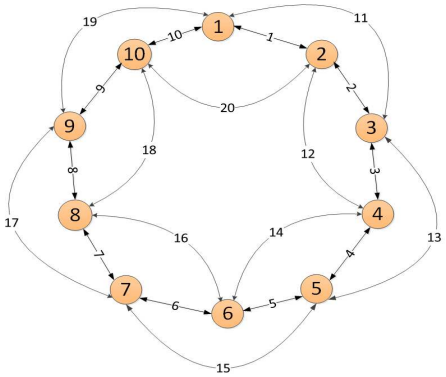

FIGURE 2: 10-node Circular Lattice Network Topology

Meanwhile, the special case of $W = 1$, which is equivalent to the full wavelength conversion, is also considered.

In this paper, we consider two topologies for the OCS network simulation, i.e. 14-node NSFNET and 10-node circular lattice network. Fig. 1 displays the NSFNET topology with 14 nodes and 21 optical links installed. In total, there are 91 S-D pairs in the network, which are bi-directional. Also, 14 routing tables are prepared for the topology. Fig. 2 shows the circular lattice network topology with 10 nodes and 20 optical links installed. 45 S-D pairs exist in the network. We prepare a routing table for this topology.

To construct the routing tables, we adopt a shortest path algorithm and a random approach. For the shortest path algorithm, it helps to generate the route for each S-D pairs. For the random approach, one available route is randomly generated as first path. After that, all links involved in the first path are excluded from the network, and another available route is generated as second path using the remaining links. The above process is repeated until the maximum number of available paths is reached. For the 14-node NSFNET, 25 S-D pairs have two link-disjoint paths, 65 S-D pairs have three link-disjoint paths and 1 S-D pair has four link-disjoint paths. For the 10-node circular lattice network, 45 S-D pairs have four link-disjoint paths. For simplicity, we only choose link-disjoint paths for each S-D pair; however, this is not necessary to achieve good load balancing.

As mentioned in Section III, seven parameters are used for the simulation and the training model. These parameters are

TABLE 3: Range of the parameters

	Range of the parameters	
	14-node NSFNET	10-node circular lattice network
E	[0.1-3.5]	[12-82]
D	[0-1]	[0-10]
W	[1-5]	[80-110]
F	[5-10]	[1-4]
APL	[2.14-5.08]	[3.28-4.67]
CR	[0.33-1.86]	[0.33-0.76]
P	[1-3]	[3]

tuned to generate the datasets corresponding to our interested range of blocking probability, i.e. [10 ^{5} 10 ^{1}]. Table 3 displays the specific ranges for the parameters.

First, the values of P is in the range of [1 3]. This is due to the fact that the number of possible S-D pairs between two nodes is bounded by the topology. In our topologies, since more than half of the S-D pairs have three paths, we consider that the value of P is bounded above by three. Second, the ranges for E and D are the empirical settings so that the dataset has our blocking probability range of interest. Third, the values of W and F are set based on the scale of the network. Finally, the values of APL and CR are calculated based on the routing tables.

B. DATASET PRE-PROCESSING AND TRAINING SETTINGS

In general, pre-processing of the dataset is required before the training stage. In this section, we shall use the term attribute to refer to the input parameters. To avoid the occurrence of outliers, each attribute is normalized by standard deviation. Afterwards, we separate the dataset into three classes based on the blocking probability as shown in Table 4. For the 14-node NSFNET topology, 10000 data entries are selected for the training set and test set in accordance with Table 4. For the 10-node circular lattices network topology, 1000 data entries are provided for the training set and test set as shown in Table 4. This approach is intended to ensure good approximation ability of the neural network across a wide range of blocking probability.

In the training stage, the Log-EEM-ELM algorithm in Section III is used to train the neural network. For the Log-EEM-ELM empirical setting, the parameter j is set to 5, and the parameter L_n is set to 10. It means that five candidate sets of ten hidden nodes are provided in the selection phase for each learning step. While $j = 10$ is also considered, the resulting neural network does not show a significant reduction in MSE despite the approximate doubling of the training time required. When the Log-EEM-ELM algorithm is completed, a number of hidden nodes with the trained output weights are generated. We then use the ADMM optimization to further remove some unimportant nodes. For the ADMM empirical setting, the parameter is set to 2, and the parameter is in the range [10 ^{4} 10 ^{1}]. The chosen range for can cover many possible solutions for the optimization.

Furthermore, two metrics are used to measure the performance of the algorithm. The first one is the MSE of the blocking probability, denoted as

$$\text{MSE} = \frac{1}{N} \sum_{k=1}^{N} (p_k \quad \hat{p}_k)^2; \tag{24}$$

where $\hat{p}_k$'s and p_k's are the estimated probability values and the true probability values, respectively. Due to the wide dynamic range for blocking probability, as mentioned in [12], the MSE might not be a good measurement for the approximation of probability values. Hence, we also consider the relative error, denoted as

$$\text{relative error} = \frac{1}{N} \sum_{k=1}^{N} \frac{|p_k \quad \hat{p}_k|}{p_k}. \tag{25}$$

Hence, we conduct an all-rounded analysis for the performance of our algorithm.

V. RESULTS AND ANALYSIS

A. COMPARISON BETWEEN LOG-I-ELM AND LOG-EEM-ELM

To provide a detailed analysis, we compare the performance of our proposed Log-EEM-ELM algorithm to that of the Log-I-ELM algorithm in [12]. The results obtained in different stages will be presented and explained in the following subsections.

To start with, the two algorithms (i.e. Log-I-ELM and Log-EEM-ELM) are used to build two SLFNs for estimating the blocking probability. Fig. 3 shows the MSE of the estimated blocking probability for the test set versus the number of hidden nodes. The Log-I-ELM algorithm gives an MSE of 0.09866 using 2951 hidden nodes, while the Log-EEM-ELM algorithm gives an MSE of 0.0006198 using 2570 hidden nodes. In other words, the test set MSE of Log-EEM-ELM is over 100 times smaller than that of Log-EEM-ELM while using 13% fewer hidden nodes.

In addition, Fig. 4 shows the relative errors of the estimated blocking probability for test set versus the number of hidden nodes. The relative error of Log-I-ELM is 5.637 when 2951 hidden nodes are used. It is over 10 times larger than that of Log-EEM-ELM, which is 0.2661 when 2570 hidden nodes are used. Considering both metrics, the Log-EEM-ELM algorithm uses fewer hidden nodes to produce more accurate results compared to the Log-I-ELM algorithm. This phenomenon is due to the selection process of hidden nodes and the updating of existing output weights in the Log-EEM-ELM algorithm. These properties ensure that the selected hidden nodes contribute to the greatest reduction of error at each incremental stage, whereas the Log-I-ELM algorithm does not incorporate this feature.

In summary, the approximation ability of the Log-EEM-ELM algorithm is stronger than that of the Log-I-ELM algorithm. In the next section, we use the ADMM framework to remove some unimportant nodes in the two trained SLFNs.

TABLE 4: CLASSES OF DATASET

		Classes		
		1E-5-1E-3	1E-3-1E-1	1E-1-0.3
14-node NSFNET	# Entries for training	2000	5000	3000
	# Entries for test	2000	5000	3000
10-node circular lattice network	# Entries for training	0	1000	0
	# Entries for test	0	1000	0

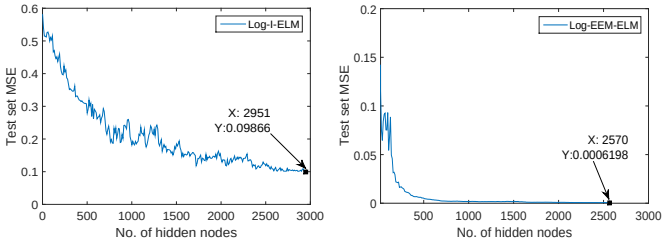

FIGURE 3: Test set MSE versus number of hidden nodes of Log-I-ELM and Log-EEM-ELM

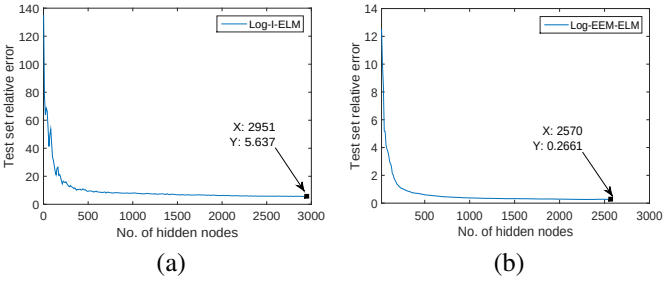

FIGURE 4: Test set relative error versus no. of hidden nodes of Log-I-ELM and Log-EEM-ELM

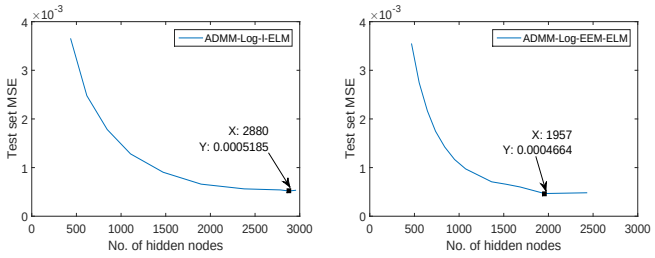

FIGURE 5: MSE versus number of hidden nodes of ADMM-Log-I-ELM and ADMM-Log-EEM-ELM

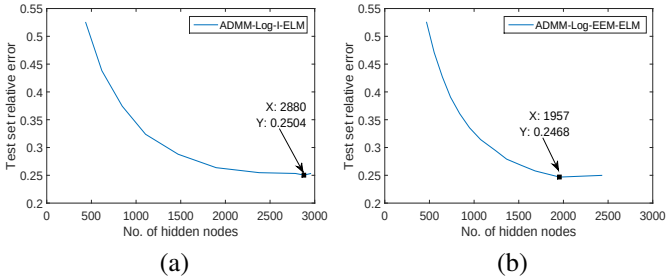

FIGURE 6: Relative error versus number of hidden nodes of ADMM-Log-I-ELM and ADMM-Log-EEM-ELM

B. COMPARISON BETWEEN ADMM-LOG-I-ELM AND ADMM-LOG-EEM-ELM

During the ADMM optimization, we can control the number of hidden nodes by varying the weighting factor . Fig. 5 presents the test set MSE versus the number of hidden nodes for the ADMM-Log-I-ELM and the ADMM-Log-EEM-ELM algorithms. For the ADMM-Log-I-ELM algorithm, 2880 hidden nodes are required to achieve the minimum test set MSE of 0.0005185. For the ADMM-Log-EEM-ELM algorithm, 1957 hidden nodes are needed to achieve the minimum test set MSE of 0.0004664. Note that the ADMM reduces both the MSE and number of hidden nodes compared to the Log-EEM-ELM and Log-I-ELM algorithms. Again, the ADMM-Log-EEM-ELM algorithm achieves a smaller MSE value and requires fewer hidden nodes than the ADMM-Log-I-ELM algorithm.

Besides that, the relative errors for both algorithms are presented in Fig. 6. The ADMM-Log-I-ELM algorithm uses 2880 hidden nodes to achieve the minimum relative error of 0.2504. Note that, compared to the Log-I-ELM, the relative error of the ADMM-Log-I-ELM algorithm drops from 5.637 to 0.2504, and the required number of hidden nodes is reduced from 2951 to 2880. In addition, the ADMM-Log-EEM-ELM algorithm uses 1957 hidden nodes to achieve the minimum relative error of 0.2468. Compared to the ADMM-Log-I-ELM, the ADMM-Log-EEM-ELM algorithm produces a lower relative error while using a third fewer hidden nodes.

On the other hand, the computational complexity for each iteration of the ADMM-Log-EEM-ELM algorithm, i.e. $O(n^2N) + O(n^2)$, is higher than that of the ADMM-Log-I-ELM algorithm, i.e. $O(N) + O(n^2)$. However, as the ADMM-Log-EEM-ELM algorithm uses fewer hidden nodes (n), the actual complexity is reduced. Also, the training time for the ADMM-Log-EEM-ELM and the ADMM-Log-I-ELM algorithms are 273 and 206 min, respectively. Like the other neural network approach, the estimation of blocking probability takes only a second to complete. In the ADMM-Log-EEM-ELM algorithm, the benefit of reducing the relative error and the required number of hidden nodes should outweigh the raise of the complexity and the training time.

In short, the ADMM-Log-EEM-ELM algorithm reduces

MSE, relative error, and the required number of hidden nodes, compared to the ADMM-Log-I-ELM. In other words, the ADMM-Log-EEM-ELM algorithm is more suitable for the estimation of blocking probability in OCS networks.

C. ESTIMATION OF BLOCKING PROBABILITY

With the ADMM optimization, the trained SLFNs can be used for estimating the blocking probability. We test the accuracy of our algorithm with respect to certain input parameters. In particular, the chosen input parameters provide a wide range of blocking probability estimation for OCS networks with fixed-alternate routing, i.e. [10 ^{5} 10 ^{1}].

Fig. 7 shows the estimated blocking probability versus the traffic load E for the 14-node NSFNET topology. Under this setting, the traffic load E is within [0:45 1:4]. Meanwhile, the other six inputs are constant, with $D = 0{:}2$, $W = 4$, $C = 5$, $P = 2$, APL = $f3{:}76; 2{:}15; 0g$ and CR = $f0{:}714; 0{:}476; 0g$. As shown in Fig. 7, most of the blocking probability estimates produced by our ADMM-Log-EEM-ELM algorithm are within the 95% confidence interval of simulation. For instance, when E is 0.45 Erlang per S-D pair, the estimate 1:788 10 ^{5} is within the 95% confidence interval of simulation, i.e. [1:426 10 ^{5}-1:908 10 ^{5}]. Similar results are obtained in Fig. 8 - Fig. 11.

Fig. 12 shows the estimated blocking probability versus the traffic load E for the 10-node circular lattice network topology. The traffic load E is in the range of [41 63] with $D = 2$, $W = 90$, $F = 3$, $P = 3$, APL = $f3{:}33; 4{:}67; 3{:}27g$ and CR = $f0{:}429; 0{:}333; 0{:}762g$. Many blocking probability estimates produced by the ADMM-Log-EEM-ELM algorithm are within the 95% confidence interval of simulation. In particular, when E =41 Erlang per S-D pair, the estimate 8:403 10 ^{5} is within the 95% confidence interval of simulation, i.e. [4:995 10 ^{5}-8:506 10 ^{5}]. Similar results are obtained in Fig. 13- Fig. 14.

In general, the ADMM-Log-EEM-ELM algorithm outperforms the ADMM-Log-I-ELM algorithm using fewer hidden nodes, due to the benefits from the selection process for candidate nodes during training.

D. COMPARISON WITH EFPA

As mentioned, the EFPA is a well-known approximation method for estimating the blocking probability in a network. In Fig. 7 - Fig. 14, we compare our ADMM-Log-EEM-ELM algorithm with the EFPA.

As the results shown, the EFPA overestimates the blocking probability of OCS networks by hundreds of times compared to the simulated results, and consistently overestimates the blocking probability across the entire range of offered loads under consideration. For instance, in Fig. 9, when E =1.5 Erlang per S-D pair, the blocking probability estimated by the EFPA is 0:00741, whereas the blocking probability estimated by the ADMM-Log-EEM-ELM is 2:821 10 ^{6}, which is within the 95% confidence interval of simulation.

As mentioned in Section II, the EFPA includes simplifying assumptions, which may result in significant errors. For the

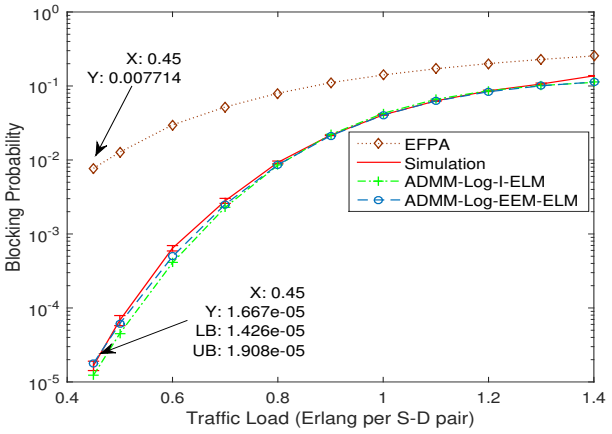

FIGURE 7: 14-node NSFNET blocking probability estimation by ADMM-Log-EEM-ELM, ADMM-Log-I-ELM and EFPA with setting D=0.2, W=4, F=5, P=2, APL={3.76, 2.15, 0}, CR={0.714, 0.476, 0}

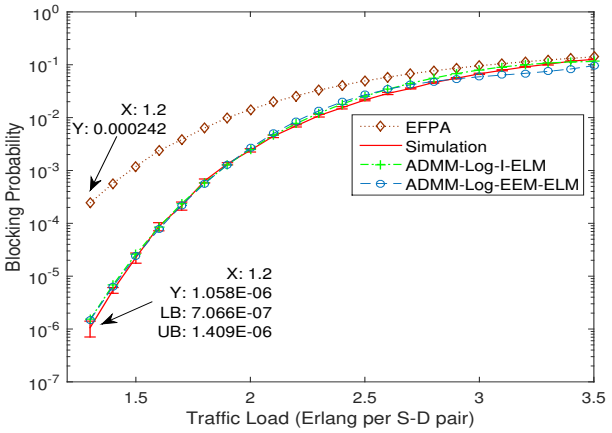

FIGURE 8: 14-node NSFNET blocking probability estimation by ADMM-Log-EEM-ELM, ADMM-Log-I-ELM and EFPA with setting D=0.3, W=5, F=9, P=3, APL={3.76, 2.15, 4.72}, CR={0.714, 0.476, 0.00119}

EFPA as applied to an OCS network, a new connection request is assumed to randomly choose a wavelength for the transmission. If the chosen wavelength is not free in the intermediate nodes, even if there are other free wavelengths, the request will be blocked. This differs from the actual behavior of OCS networks, in which the request may use any free wavelength instead. Due to this assumption, the EFPA overestimates the blocking probability of OCS networks with fixed-alternate routing.

Moreover, the computational complexity of the EFPA is $O(\ WE^{F})$ for each iteration, where and E are the number of links in the topology and the mean traffic load for all S-D pairs, respectively. Although the complexity of EFPA is simpler than that of our proposed algorithms, the accuracy and the convergence of EFPA are not guaranteed, as shown in Fig. 7 - Fig. 14. This explains that the ADMM-Log-EEM-ELM algorithm is more suitable for estimating the blocking probability in the OCS networks with fixed-alternate routing.

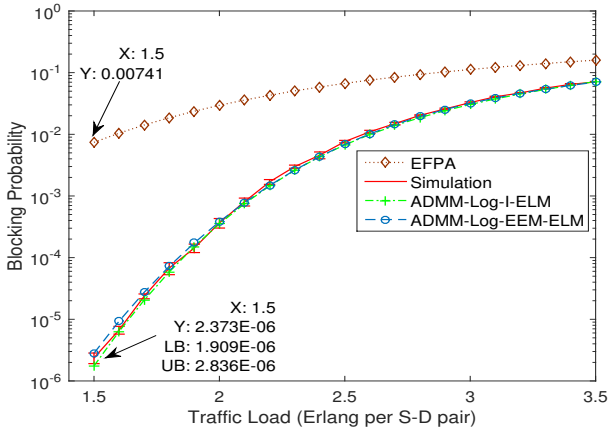

FIGURE 9: 14-node NSFNET blocking probability estimation by ADMM-Log-EEM-ELM, ADMM-Log-I-ELM and EFPA with setting D=0.1, W=5, F=10, P=1, APL={2.25, 0, 0}, CR={0.571, 0, 0}

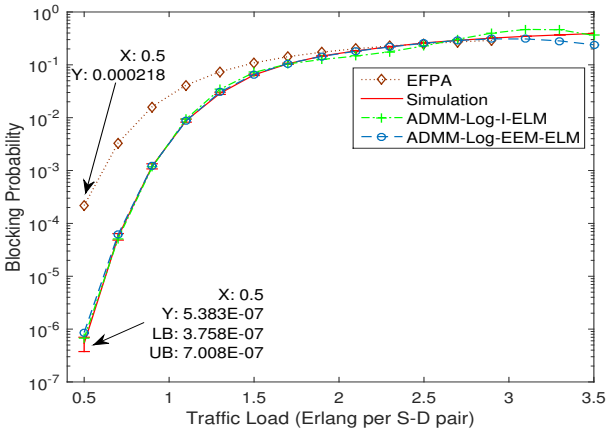

FIGURE 11: 14-node NSFNET blocking probability estimation by ADMM-Log-EEM-ELM, ADMM-Log-I-ELM and EFPA with setting D=0.2, W=5, F=5, P=3, APL={3.769, 2.153, 4.727}, CR={0.714, 0.476, 1.048}

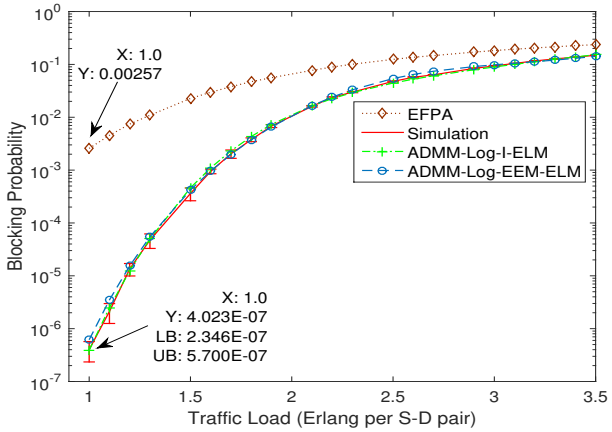

FIGURE 10: 14-node NSFNET blocking probability estimation by ADMM-Log-EEM-ELM, ADMM-Log-I-ELM and EFPA with setting D=0.5, W=4, F=10, P=1, APL={2.25, 0, 0}, CR={0.571, 0, 0}

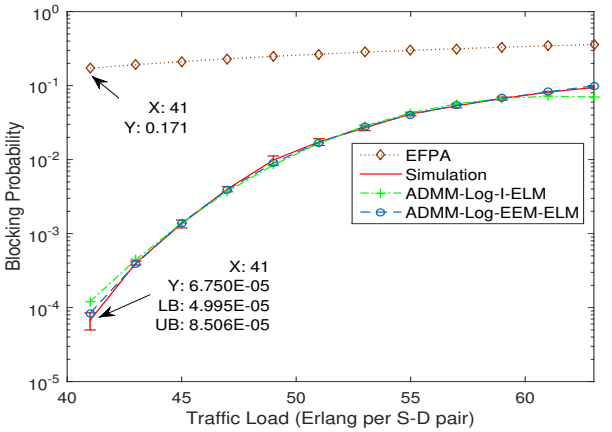

FIGURE 12: 10-node circular lattice network blocking probability estimation by ADMM-Log-EEM-ELM, ADMM-Log-I-ELM and EFPA with setting D=2, W=90, F=3, P=3, APL={3.33, 4.67, 3.27}, CR={0.429, 0.333, 0.762}

VI. CONCLUSION

In this paper, we propose an enhancement of ELM model, namely ADMM-Log-EEM-ELM, to estimate the blocking probability in OCS networks with fixed-alternate routing. For the ADMM-Log-EEM-ELM algorithm, the EEM-ELM is used to train a neural network, using the logarithm of the blocking probability as the target outputs, before the ADMM optimization is applied to reduce the required number of nodes in the network. To demonstrate the performance of our algorithm, we compare it to the ADMM-Log-I-ELM algorithm, which was proposed in our previous paper. Numerical results demonstrate that our ADMM-Log-EEM-ELM algorithm outperforms the ADMM-Log-I-ELM algorithm in terms of MSE, relative error and the required number of hidden nodes. The estimation times for our algorithm are within a second, which is faster than the 20 minutes runtime required by computer simulation. Finally, we compare the performance of the ADMM-Log-EEM-ELM algorithm to that of the well-known analytical approximation EFPA. Numerical results demonstrate that our ADMM-Log-I-ELM algorithm is hundreds of times more accurate than the EFPA. In summary, we demonstrate the feasibility of enhancement of ELM approach on blocking probability estimation for OCS networks under fixed-alternate routing scheme. We are confident that our enhancement of ELM framework can also be applied to other kinds of networks with different routing schemes, which will be our future work.

REFERENCES

[1] A. Girard, Routing and dimensioning in circuit-switched networks. Addison-Wesley Longman Publishing Co., Inc., 1990.
[2] A. Inoue, "An advanced large-scale simulation system for telecommunication networks with dynamic routing," NETWORKS'89, pp. 77–82, 1989.
[3] F. P. Kelly, "Blocking probabilities in large circuit-switched networks," Advances in applied probability, vol. 18, no. 2, pp. 473–505, 1986.
[4] H. Rauthan, A. Verma, and G. Kaur, "Congestion control strategy in optical burst switching networks," International Journal of Computer Applications, vol. 91, no. 17, pp. 33–37, 2014.

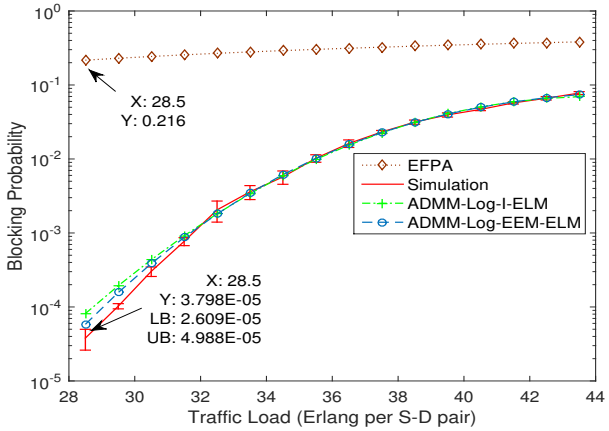

FIGURE 13: 10-node circular lattice network blocking probability estimation by ADMM-Log-EEM-ELM, ADMM-Log-I-ELM and EFPA with setting D=4, W=100, F=2, P=3, APL={3.33, 4.67, 3.27}, CR={0.429, 0.333, 0.762}

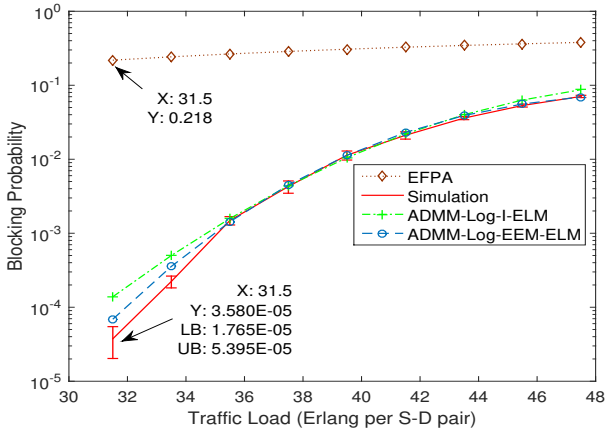

FIGURE 14: 10-node circular lattice network blocking probability estimation by ADMM-Log-EEM-ELM, ADMM-Log-I-ELM and EFPA with setting D=2, W=110, F=2, P=3, APL={3.33, 4.67, 3.27}, CR={0.429, 0.333, 0.762}

[5] M. Wang, S. Li, E. W. Wong, and M. Zukerman, "Performance analysis of circuit switched multi-service multi-rate networks with alternative routing," Journal of Lightwave Technology, vol. 32, no. 2, pp. 179–200, 2014.

[6] A. Birman, "Computing approximate blocking probabilities for a class of all-optical networks," IEEE Journal on Selected areas in Communications, vol. 14, no. 5, pp. 852–857, 1996.

[7] F. P. Kelly, "Blocking probabilities in large circuit-switched networks," The Annals of Applied Probability, vol. 18, no. 2, pp. 473–505, Jun. 1986.

[8] G. Huang, L. Chen, and C. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," IEEE Transactions on Neural Networks, vol. 17, no. 4, pp. 879–892, July 2006.

[9] J. Tang, C. Deng, and G. B. Huang, "Extreme learning machine for multilayer perceptron," IEEE Transactions on Neural Networks and Learning Systems, vol. 27, no. 4, pp. 809–821, April 2016.

[10] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," Neurocomputing, vol. 70, no. 1, pp. 489–501, 2006.

[11] Y. Lan, Y. C. Soh, and G.-B. Huang, "Random search enhancement of error minimized extreme learning machine." in ESANN, 2010.

[12] H. C. Leung, C. S. Leung, E. W. Wong, and S. Li, "Extreme learning machine for estimating blocking probability of bufferless OBS/OPS networks," Journal of Optical Communications and Networking, vol. 9, no. 8, pp. 682–692, 2017.

[13] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," Found. Trends Mach. Learn., vol. 3, no. 1, pp. 1–122, Jan. 2011.

[14] M. V. Afonso, J. M. Bioucas-Dias, and M. A. Figueiredo, "An augmented lagrangian approach to the constrained optimization formulation of imaging inverse problems," IEEE Trans. Image Process., vol. 20, no. 3, pp. 681–695, Mar. 2011.

[15] H. Harai, M. Murata, and H. Miyahara, "Performance of alternate routing methods in all-optical switching networks," in INFOCOM'97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., Proceedings IEEE, vol. 2. IEEE, 1997, pp. 516–524.

[16] B. Karanov, M. Chagnon, F. Thouin, T. A. Eriksson, H. Bülow, D. Lavery, P. Bayvel, and L. Schmalen, "End-to-end deep learning of optical fiber communications," arXiv preprint arXiv:1804.04097, 2018.

[17] N. Merayo, D. Juárez, J. C. Aguado, I. De Miguel, R. Durán, P. Fernández, R. M. Lorenzo, and E. Abril, "Pid controller based on a self-adaptive neural network to ensure qos bandwidth requirements in passive optical networks," Journal of Optical Communications and Networking, vol. 9, no. 5, pp. 433–445, 2017.

[18] W. Mo, C. L. Gutterman, Y. Li, S. Zhu, G. Zussman, and D. C. Kilper, "Deep-neural-network-based wavelength selection and switching in roadm systems," Journal of Optical Communications and Networking, vol. 10, no. 10, pp. D1–D11, 2018.

[19] D. R. B. de Araujo, C. J. A. Bastos-filho, and J. F. Martins-filho, "Methodology to obtain a fast and accurate estimator for blocking probability of optical networks," IEEE/OSA Journal of Optical Communications and Networking, vol. 7, no. 5, pp. 380–391, May 2015.

[20] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," Neural Networks, vol. 2, no. 5, pp. 359 – 366, 1989.

[21] K. Hornik, "Approximation capabilities of multilayer feedforward networks," Neural Networks, vol. 4, no. 2, pp. 251 – 257, 1991.

[22] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," IEEE Transactions on Information Theory, vol. 39, no. 3, pp. 930–945, May 1993.

[23] X. Liu, S. Lin, J. Fang, and Z. Xu, "Is extreme learning machine feasible? a theoretical assessment (part i)," IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 1, pp. 7–20, 2015.

[24] S. Lin, X. Liu, J. Fang, and Z. Xu, "Is extreme learning machine feasible? a theoretical assessment (part ii)," IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 1, pp. 21–34, 2015.

[25] H. T. Huynh and Y. Won, "Extreme learning machine with fuzzy activation function," in INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on. IEEE, 2009, pp. 303–307.

[26] G. Feng, G.-B. Huang, Q. Lin, and R. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," IEEE Transactions on Neural Networks, vol. 20, no. 8, pp. 1352–1357, 2009.

[27] R. Cooper and S. Katz, "Analysis of alternate routing networks with account taken of nonrandomness of overflow traffic," Memo., Bell Telephone Lab, 1964.

[28] E. W. Wong, A. Zalesky, Z. Rosberg, and M. Zukerman, "A new method for approximating blocking probability in overflow loss networks," Computer Networks, vol. 51, no. 11, pp. 2958–2975, 2007.

[29] M. Wang, S. Li, E. W. Wong, and M. Zukerman, "Blocking probability analysis of circuit-switched networks with long-lived and short-lived connections," Journal of Optical Communications and Networking, vol. 5, no. 6, pp. 621–640, 2013.

[30] W. Whitt, "Blocking when service is required from several facilities simultaneously," Bell Labs Technical Journal, vol. 64, no. 8, pp. 1807–1856, 1985.

[31] R. Lao and R. Killey, "Design of wavelength-routed optical network topologies to minimise lightpath blocking probabilities," Optical Networks Group, Department of Electronic and Electrical Engineering, 2003.

[32] R. A. Barry and P. A. Humblet, "Models of blocking probability in all-optical networks with and without wavelength changers," in INFOCOM'95. Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Bringing Information to People. Proceedings. IEEE, vol. 2. IEEE, 1995, pp. 402–412.

[33] S. Baroni, P. Bayvel, R. J. Gibbens, and S. K. Korotky, "Analysis and design of resilient multifiber wavelength-routed optical transport networks," Journal of lightwave technology, vol. 17, no. 5, pp. 743–758, 1999.

SHUO LI received the B.S degree from City University of Hong Kong, Hong Kong SAR in 2009 and the Ph.D. from the same University in 2014. She is currently a Lecture in the School of Engineering, RMIT University, Australia. Her research interests include telecommunication, analysis and design of optical networks and core networks.

HO CHUN LEUNG received his B.Eng. degree in Information Engineering from City University of Hong Kong, Hong Kong SAR, China, in 2015. He is currently pursuing his PhD degree at Department of Electronic Engineering, City University of Hong Kong. His research interest includes machine learning, neural network, computer Graphic, and telecommunication.

ERIC WING MING WONG (S'87–M'90–SM'00) received the B.Sc. and M.Phil. degrees in electronic engineering from the Chinese University of Hong Kong, Hong Kong, in 1988 and 1990, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Massachusetts, Amherst, MA, USA, in 1994. He is an Associate Professor with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong. His research interests include analysis and design of telecommunications and computer networks, energy-efficient data center design, green cellular networks and optical networking.

CHI SING LEUNG (M'05–SM'15) received the PhD. degree in computer science from the Chinese University of Hong Kong in 1995. He is currently a Professor in the Department of Electronic Engineering, City University of Hong Kong. His research interests include neural computing and computer graphics. He has published over 120 journal papers in the areas of Digital Signal Processing, Neural Networks, and Computer Graphics. In 2005, he received the 2005 IEEE Transactions on Multimedia Prize Paper Award for his paper titled, "The Plenoptic Illumination Function". He was a member of Organizing Committee of ICONIP2006. He was the Program Chair of ICONIP2009 and ICONIP2012. He is/was the guest editors of several journals, including Neural Computing and Applications, Neurocomputing, and Neural Processing Letters. He is a governing board member of the Asian Pacific Neural Network Assembly (APNNA) and Vice President of APNNA.