
SHUO LI1, (Member, IEEE), HO CHUN LEUNG2, ERIC WING MING WONG2, (Senior Member, IEEE), AND CHI SING LEUNG2, (Senior Member, IEEE)

1School of Engineering, RMIT University, Melbourne, VIC 3000, Australia
2Department of Electronic Engineering, City University of Hong Kong, Hong Kong

Corresponding author: Chi Sing Leung (eeleungc@cityu.edu.hk)

This work was supported in part by the Research, Hong Kong, under Grant CityU 11259516, and in part by the City University of Hong Kong under Project 7004842 and Project 7005063.

ABSTRACT In 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. The numerical results showed that the neural-network-based estimation was accurate and thousands of times faster than a 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. The 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 with 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-known 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) [3]. 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 [7]–[9]. The enhancement of error minimized ELM (EEM-ELM) algorithm [10] 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 [11], 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) [12], [13] 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], [14]. 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 modeling in an optical network by embedding a deep fully-connected feed-forward neural network [15]. A significant improvement was observed in the intensity modulation direct detection (IM-DD) system. Merayo et al. [16] 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. [17] 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. [18] introduced a multilayer perceptron (MLP) model, a classic neural network approach, to estimate the blocking probability of an OCS network. In our previous paper [11], 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 \( \mathcal{D}_t = \{ (x_l, p_l) : x_l \in \mathbb{R}^d, p_l \in \mathbb{R}, l = 1, \ldots, N \} \), where \( x_l \) and \( p_l \) are the input vector and target output of the \( l \)-th sample, respectively. In [19]–[21], the network output of the SLFN approach is expressed as

\[
f_n(x) = \sum_{i=1}^{n} \beta_i g_i(x)
\]

(1)

where \( g_i(x) \) is the output of the \( i \)-th hidden node, and \( \beta_i \) is the weight between the \( i \)-th hidden node and the output node.

For the hidden nodes in the ELM, the activation functions are defined as sigmoid functions \([22]–[24]\). In other words, the output of the \( i \)-th hidden node is expressed as

\[
g_i(x) = \frac{1}{1 + \exp(-a_i^T x + b_i)}
\]

(2)

where \( a_i \) is the input weight vector of the \( i \)-th hidden node, and \( b_i \) is the bias term of the \( i \)-th 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 \([7], [8]\). When \( n \) hidden nodes are used for approximation, the MSE is expressed as

\[
\mathcal{E} = \sum_{l=1}^{N} (p_l - \sum_{i=1}^{n} \beta_i g_i(x_l))^2 = \|p - \sum_{i=1}^{n} \beta_i g_i\|_2^2
\]

(3)

where \( p = [p_1, \ldots, p_N]^T \), and \( g_i = [g_i(x_1), \ldots, g_i(x_N)]^T \).

In [25], 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:

\[
\mathcal{E}_n = \|p - \Phi_n \beta\|_2^2,
\]

(4)

\[
\Phi_n = \begin{pmatrix}
g_1(x_1) & \cdots & g_n(x_1) \\
g_1(x_2) & \cdots & g_n(x_2) \\
\vdots & \ddots & \vdots \\
g_1(x_N) & \cdots & g_n(x_N)
\end{pmatrix}
\]

(5)

The hidden layer matrix \( \Phi_{n+1} \) can be further expressed as \( \Phi_{n+1} = [\Phi_n, \delta \Phi_n] \).

At the \( n \)-th incremental learning step, a group of new nodes \( \delta \Phi_n \) is generated randomly, and added into the network. For the newly added nodes, we need to update the output weights \( \beta_n \). The calculation of output weights is formulated as follows:

\[
Q_n = ((I - \Phi_n \Phi_n^T)\delta \Phi_n)^T
\]

(6)

\[
T_n = \Phi_{n+1}^T (I - \delta \Phi_n Q_n)
\]

(7)

\[
\beta_{n+1} = \begin{bmatrix}
T_n \\
Q_n
\end{bmatrix} p
\]

(8)

where \( \dagger \) denotes the Moore-Penrose inverse.

Hereafter, Lan et al. [10] 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 [12], [13], the authors proposed the ADMM framework to solve the convex optimization problem using an iterative approach. The constraint optimization problem is shown as follows:

\[
\min_{\beta, u} \psi(\beta) + \phi(u) \\
\text{s.t. } C\beta + Du = v,
\]

(9)

where \( \psi \) and \( \phi \) are the cost terms in the objective function, and \( \beta \) and \( u \) are decision variable vectors in the optimization.

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

\[
L(\beta, u, y) = \psi(\beta) + \phi(u) + y^T(C\beta + Du - v) + \frac{\rho}{2}\|C\beta + Du - v\|_2^2,
\]

(10)

where \( \rho \) is a positive penalty parameter, and \( y \) is the Lagrangian vector. Hence, the iterative process for \( (\beta, u, y) \) is given by

\[
\beta^{k+1} = \arg \min_{\beta} L(\beta, u^k, y^k),
\]

(11a)

\[
u^{k+1} = \arg \min_{u} L(\beta^{k+1}, u, y^k),
\]

(11b)

\[
y^{k+1} = y^k + \rho(C\beta^{k+1} + Du^{k+1} - v).
\]

(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 [26]. 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 [27], [28] 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 [3], [29]. 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 \( \{x_i : x_i \in \mathbb{R}^d\} \), the corresponding blocking probability \( \{p_i : p_i \in \mathbb{R}_{>0}\} \) 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 [11] illustrates this problem. Therefore, in [11], we proposed to train our SLFN using the logarithm of the blocking probability, \( \{\tilde{p}_i = \log p_i\} \), instead of \( p \). The new training set is denoted as:

\[
\tilde{\mathbb{D}} = \left\{(x_i, \tilde{p}_i = \log p_i) : x_i \in \mathbb{R}^d, \tilde{p}_i \in \mathbb{R}_{>0}, i = 1, \cdots , N\right\}.
\]

The training objective becomes

\[
\tilde{E} = \left\| \tilde{p} - \Phi \beta \right\|_2^2,
\]

where \( \tilde{p} = [\tilde{p}_1, \cdots , \tilde{p}_N]^T \).

Based on the EEM-ELM, we develop a new ELM algorithm, namely Log-EEM-ELM. For minimizing the objective function \( \tilde{E} \), 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. \( \{g_1(x), \cdots , g_{n_e}(x)\} \). 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

\[
\tilde{E} = \left\| \tilde{p} - \sum_{i=1}^{n_e} \beta_i g_i \right\|_2^2.
\]

To remove the unimportant nodes, an \( \ell_1 \) norm penalty term is required in the objective function. The modified objective function is denoted as

\[
\tilde{J} = \frac{1}{2} \left\| \tilde{p} - \Phi \beta \right\|_2^2 + \lambda \|\beta\|_1,
\]

where \( \lambda \) is a tradeoff parameter between the number of selected nodes and the approximation error of the log blocking probability. \( \lambda \) can be tuned to trade off the number of hidden nodes with the approximation accuracy. When \( \lambda \) is large, the SLFN contains fewer hidden nodes. When \( \lambda \) 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 \( u \) is introduced, such that the optimization problem becomes a constrained one. The formulation of the problem becomes

\[
\min \beta, u \quad \frac{1}{2} \left\| \tilde{p} - \Phi \beta \right\|_2^2 + \lambda \|u\|_1,
\]

s.t. \( u = \beta \).
The augmented Lagrangian of (16) is denoted as
\[
\mathcal{L}(\beta, u, \gamma) = \frac{1}{2} \| \bar{p} - \Phi \beta \|_2^2 + \lambda \| u \|_1 \\
+ \gamma^T (u - \beta) + \frac{\rho}{2} \| u - \beta \|_2^2.
\]  
(17)

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

**Computing \( \beta^{k+1} \):**

By fixing \( u \) as \( u^k \) and \( \gamma \) as \( \gamma^k \) in (17), \( \beta^{k+1} \) can be updated by
\[
\beta^{k+1} = \arg \min_{\beta} \mathcal{L}(\beta, u^k, \gamma^k) \\
= \arg \min_{\beta} \frac{1}{2} \| \bar{p} - \Phi \beta \|_2^2 + \gamma^T (u^k - \beta) + \frac{\rho}{2} \| u^k - \beta \|_2^2 \\
= \arg \min_{\beta} \frac{1}{2} \| \bar{p} - \Phi \beta \|_2^2 + \frac{\rho}{2} \| u^k - \beta \|_2^2 + \frac{1}{\rho} \gamma^k. 
\]  
(18)

From (18), we obtain
\[
\beta^{k+1} = \left[ \Phi^T \Phi + \rho I \right]^{-1} \left[ \Phi^T \bar{p} + \rho u^k + \gamma^k \right].
\]  
(19)

where \( I \) is an identity matrix.

**Computing \( u^{k+1} \):**

By fixing \( \beta \) as \( \beta^{k+1} \) and \( \gamma \) as \( \gamma^k \) in (17), \( u^{k+1} \) can be updated by
\[
u^{k+1} = \arg \min_u \lambda \| u \|_1 + \frac{\rho}{2} \| u - \beta^{k+1} \|_2 + \frac{1}{\rho} \gamma^k. 
\]  
(20)

The term \( \lambda \| u \|_1 \) in (20) is non-differentiable; however, the closed-form solution can be obtained by using sub-differential calculus [13]. The formulation is denoted as
\[
u^{k+1} = S_{\lambda/\rho} \left( \beta^{k+1} - \frac{1}{\rho} \gamma^k \right).
\]  
(21)

where \( S_{\cdot}(\cdot) \) is a soft-threshold operator [13], given by
\[
S_{\cdot}(\cdot) = \operatorname{sgn}(\cdot) \cdot (|\cdot| - \zeta).
\]  
(22)

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

**Computing \( \gamma^{k+1} \):**

\[
\gamma^{k+1} = \gamma^k + \rho \left( u^{k+1} - \beta^{k+1} \right).
\]  
(23)

In short, we notice that the computational complexity of ADMM-Log-EEM-ELM for each iteration is \( O(n^2N) + O(n^3) \).

**B. SIMULATION AND TRAINING PARAMETERS**

For the neural network training inputs, we consider seven parameters for an OCS network, as defined in Table 1. They are similar to our previous paper [11], 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 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 \([E - D/2, E + D/2]\). 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 [30]. The value \( CR_i \), as defined in Araujo et al. [18], 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>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( E )</td>
<td>Mean traffic load per S-D pair (Erlang)</td>
</tr>
<tr>
<td>( D )</td>
<td>Difference between max and min traffic load per S-D pair</td>
</tr>
<tr>
<td>( W )</td>
<td>No. of wavelengths per fiber</td>
</tr>
<tr>
<td>( F )</td>
<td>No. of fiber pairs per link</td>
</tr>
<tr>
<td>( APL )</td>
<td>Average path length in the three layers</td>
</tr>
<tr>
<td>( CR )</td>
<td>{ ( CR_1 ), ( CR_2 ), ( CR_3 ) } in the three layers</td>
</tr>
<tr>
<td>( P )</td>
<td>No. of allowable paths</td>
</tr>
</tbody>
</table>

| Table 1. Independent input parameters. |
$P$, a network allows a connection request to overflow $P-1$ times. In fact, an S-D pair can use $\min(P, R_m)$ paths, where $R_m$ 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 $\text{APL}_3 > \text{APL}_2 > \text{APL}_1$, increasing $P$ might force the overflowed connection requests to use longer paths, hence the blocking probability would be increased.

In our previous work [11], the significance of $E, D, W, F$ (the same meaning as $C$ in [11]), $\text{APL}$ and $\text{CR}$ 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 $\text{APL}$ and $\text{CR}$ are transformed into vector form $\{\text{APL}_1, \text{APL}_2, \text{APL}_3\}$ and $\{\text{CR}_1, \text{CR}_2, \text{CR}_3\}$.

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 $1.491 \times 10^{-3}$ to $6.743 \times 10^{-4}$. With the additional layer information for $\text{APL}_2$ and $\text{CR}_2$, the MSE further reduces to $5.879 \times 10^{-4}$. With additional information for $\text{APL}_3$ and $\text{CR}_3$, the MSE further reduces to $4.664 \times 10^{-4}$. These results demonstrate that these new input parameters are essential for our training model to provide an accurate estimation of blocking probability.

**TABLE 2. Trained models with various parameters.**

<table>
<thead>
<tr>
<th>Input Parameters</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E, D, W, F, {\text{CR}_1}$, ${\text{APL}_1}$</td>
<td>$1.491 \times 10^{-3}$</td>
</tr>
<tr>
<td>$E, D, W, F, {\text{CR}_1}$, ${\text{APL}_1}$</td>
<td>$6.743 \times 10^{-4}$</td>
</tr>
<tr>
<td>$E, D, W, F, {\text{CR}_1, \text{CR}_2}$, ${\text{APL}_1, \text{APL}_2}$</td>
<td>$5.879 \times 10^{-4}$</td>
</tr>
<tr>
<td>$E, D, W, F, {\text{CR}_1, \text{CR}_2, \text{CR}_3}$, ${\text{APL}_1, \text{APL}_2, \text{APL}_3}$</td>
<td>$4.664 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

**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 [31], [32], 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. 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.

**FIGURE 1. 14-node NSFNET Topology.**

**FIGURE 2. 10-node Circular Lattice Network 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 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 $\delta L_m$ 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 $\rho$ is set to 2, and the parameter $\lambda$ is in the range $[10^{-4} - 10^{-1}]$. The chosen range for $\lambda$ 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

$$MSE = \frac{1}{N} \sum_{k=1}^{N} (p_k - \hat{p}_k)^2,$$

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 [11], the MSE might not be a good measurement for the approximation of probability values. Hence, we also consider the relative error, denoted as

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

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 [11]. 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

### TABLE 3. Range of the parameters.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>14-node NSFNET</th>
<th>10-node circular lattice network</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>[0.1-3.5]</td>
<td>[12-82]</td>
</tr>
<tr>
<td>$D$</td>
<td>[0-1]</td>
<td>[0-10]</td>
</tr>
<tr>
<td>$W$</td>
<td>[1-5]</td>
<td>[80-110]</td>
</tr>
<tr>
<td>$F$</td>
<td>[5-10]</td>
<td>[1-4]</td>
</tr>
<tr>
<td>$APL$</td>
<td>[2.14-5.08]</td>
<td>[3.28-4.67]</td>
</tr>
<tr>
<td>$CR$</td>
<td>[0.33-1.86]</td>
<td>[0.33-0.76]</td>
</tr>
<tr>
<td>$P$</td>
<td>[1-3]</td>
<td>[3]</td>
</tr>
</tbody>
</table>

### TABLE 4. Classes of dataset.

<table>
<thead>
<tr>
<th>Topology</th>
<th>Classes</th>
<th>1E-5-1E-3</th>
<th>1E-3-1E-1</th>
<th>1E-1-0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>14-node NSFNET</td>
<td># Entries for training</td>
<td>2000</td>
<td>5000</td>
<td>3000</td>
</tr>
<tr>
<td></td>
<td># Entries for test</td>
<td>2000</td>
<td>5000</td>
<td>3000</td>
</tr>
<tr>
<td>10-node circular</td>
<td># Entries for training</td>
<td>0</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>lattice network</td>
<td># Entries for test</td>
<td>0</td>
<td>1000</td>
<td>0</td>
</tr>
</tbody>
</table>
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-I-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.

**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 $\lambda$. 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. $O(n^2N) + O(n^3)$, is higher than that of the ADMM-Log-I-ELM algorithm, i.e. $O(N) + O(n^3)$. 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 estimating the 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. $[10^{-5} - 10^{-1}]$.

![FIGURE 7. 14-node NSFNET blocking probability estimation by ADMM-Log-EEM-ELM, ADMM-Log-I-ELM and EFPA with setting $D = 0.2$, $W = 4$, $F = 5$, $P = 2$, $APL = [3.76, 2.15, 0]$, $CR = [0.714, 0.476, 0]$.](image)

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 $D = 0.3$, $W = 5$, $F = 9$, $P = 3$, $APL = [5.76, 2.15, 4.72]$, $CR = [0.714, 0.476, 0.00119]$.](image)

FIGURE 8. 14-node NSFNET blocking probability estimation by ADMM-Log-EEM-ELM, ADMM-Log-I-ELM and EFPA with setting $D = 0.3$, $W = 5$, $F = 9$, $P = 3$, $APL = [5.76, 2.15, 4.72]$, $CR = [0.714, 0.476, 0.00119]$.

![FIGURE 9. 14-node NSFNET blocking probability estimation by ADMM-Log-EEM-ELM, ADMM-Log-I-ELM and EFPA with setting $D = 0.1$, $W = 5$, $F = 10$, $P = 1$, $APL = [2.25, 0, 0]$, $CR = [0.371, 0, 0]$.](image)

FIGURE 9. 14-node NSFNET blocking probability estimation by ADMM-Log-EEM-ELM, ADMM-Log-I-ELM and EFPA with setting $D = 0.1$, $W = 5$, $F = 10$, $P = 1$, $APL = [2.25, 0, 0]$, $CR = [0.371, 0, 0]$.

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 $2.821 \times 10^{-6}$, 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 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 $O(\Psi W \tilde{E}^F)$ for each iteration, where $\Psi$ and $\tilde{E}$ 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.
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

HO CHUN LEUNG received the B.Eng. degree in information engineering from the City University of Hong Kong, Hong Kong, China, in 2015, where he is currently pursuing the Ph.D. degree with the Department of Electronic Engineering. His research interests include machine learning, neural networks, computer graphics, and telecommunications.

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 the 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 Ph.D. degree in computer science from the Chinese University of Hong Kong, in 1995, where he is currently a Professor with the Department of Electronic Engineering. 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 has received the 2005 IEEE Transactions on Multimedia Prize Paper Award for his paper The Plenoptic Illumination Function. He was a member of the organizing committee of ICONIP2006. He was the Program Chair of ICONIP 2009 and ICONIP 2012. He is/was a Guest Editor 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 the Vice President of APNNA.