# Extreme Learning Machine for Estimating Blocking Probability of Bufferless OBS/OPS Networks

Ho Chun Leung, Chi Sing Leung, Eric W. M. Wong, and Shuo Li

***Abstract*—Recently, the neural network approach for the blocking probability evaluation on optical networks was proposed, in which the inputs of the neural network were the optical network parameters and the output was the blocking probability of the optical network. The numerical results showed that its evaluation speed of the blocking probability was thousands of times faster than that of the discrete event simulator. However, the existing approach had two drawbacks. First, when the blocking probability was small, there was a significant approximation error due to the high dynamic range of the blocking probability. Second, the single-hidden-layer feedforward network model was used, which needed some time-consuming training algorithms to learn the parameters of hidden nodes, such as backpropagation. To solve these problems, this paper proposes to use the mean squared error of the log blocking probability as the training objective and use the extreme learning machine (ELM) framework for the training. Our numerical results show that the blocking probability estimated by our training objective is much more accurate than that of the existing approach, and it is obtained efficiently due to the greatly simplified training procedure offered by the ELM.**

***Index Terms*—Artificial neural network; Blocking probability; Network performance evaluation; Optical network.**

## I. Introduction

Optical burst switching (OBS) and optical packet switching (OPS) are the networking technologies that transmit data through optical cross connects over optical networks. In a bufferless OBS network, the control header and data are sent over two separated channels. To transmit data between a source-destination (S-D) pair, a control header is first transmitted for reserving the required bandwidth for data transmission, and a data burst is then sent without waiting for the acknowledgment. The data burst from the different S-D pairs can use the same wavelength effectively in a time-shared manner [1–3]. If a certain bandwidth in some intermediate nodes could not be reserved, the data burst would be dropped. This behavior is defined as blocking. In a bufferless OPS network, a packet that consists of the optical control header and data is sent to the other nodes. The intermediate nodes, along the path of the S-D pair, process the header and decide the output port [4]. In this case, if all channels were occupied, packet loss would occur. This packet loss is defined as blocking.

Blocking probability is defined as the ratio of the number of lost bursts/packets to the total number of sent bursts/packets. It is commonly used for the performance measurement of an OBS/OPS network. With blocking probability and the equation set proposed by [5,6], we could calculate the congestion of the network and then enable the algorithms [7] to conduct the congestion control. In other words, the evaluation of blocking probability is an important process for controlling traffic congestion. Due to the high dimensionality, the exact analysis for blocking probability is nonobtainable for a realistic size optical network. A practical way to estimate blocking probability is by computer simulation, such as discrete event simulation [8–10]. However, these methods are very time-consuming. Another practical way is by an analytical approximation, such as the Erlang fixed point approximation (EFPA) [11,12], by making some simplifying assumptions, which may make the approximation not accurate enough.

Recently, Arajo *et al.* [13] suggested a neural network approach for learning the unknown mapping from the parameters of the optical circuit switching (OCS) network to the blocking probability. The advantage of this approach [13] was that it was thousands of times faster than the discrete event simulation. However, there were two drawbacks. Since the dynamic range of the blocking probability was very large, which could vary from $10^{-5}$ to $10^{-1}$, there would be a significant approximation error when the blocking probability was very small. Therefore, using the traditional mean squared error (MSE) of the blocking probability, i.e., in the original domain, as the training objective was inappropriate. Furthermore, the existing neural approach [13] used the single-hidden-layer feedforward network (SLFN) model [14–16] for the approximation. It needed to train all the connection weights of an SLFN, including the input bias terms and the input weights of the hidden nodes. Consequently, the neural approach, proposed in Ref. [13], had a number of difficulties in the learning process, such as local minimum.

Manuscript received March 7, 2017; revised June 5, 2017; accepted June 16, 2017; published 0, 0000 (Doc. ID 290199).

H. C. Leung, C. S. Leung, and Eric W. M. Wong (e-mail: eeewong@cityu.edu.hk) are with the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong.

S. Li is with the School of Electrical and Information Engineering, Tianjin University, China.

https://doi.org/10.1364/JOCN.99.099999

To solve these problems, this paper proposes to use the MSE of the log blocking probability as the training objective. In addition, we use the extreme learning machine (ELM) framework for the training. In our approach, the approximation for the different ranges of blocking probability can then be performed in a better way, compared to the existing approach [13]. The ELM framework, in which the input biases and input weights of the hidden nodes are randomly selected, has demonstrated the ability of universal approximation [17–19]. We adopt the ELM concept to train an SLFN model for approximating the log blocking probability. Our SLFN model is constructed incrementally using an incremental learning algorithm, called incremental-ELM (I-ELM) [17]. Afterward, we use the alternating direction method of multipliers (ADMM) [20,21] to remove some unimportant nodes.

Our numerical results show that the blocking probability approximated by our training objective is much more accurate than that of the training in the original domain. In particular, for a network with a relatively light load, training in the original domain could overestimate the blocking probability by 25 times, whereas ours was within 10% error (see Section V). In addition, in our training scheme, the trained network could be obtained efficiently due to the greatly simplified training procedure offered by the ELM.

The remainder of the paper is organized as follows. Section II briefly reviews the related works. Section III introduces our proposed approaches. Section IV is the experimental 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. Bufferless OBS/OPS

In Ref. [22], in an OBS network, packets are aggregated into data bursts at the ingress node and disassembled at the egress node. For each data burst, a control header, which contains the description of the sender, is transmitted first through the control channel to reserve the wavelength for the data burst between the source and the destination so that the data burst could be transmitted all optically through data channels. Later, the data burst is transmitted without any acknowledgment by the ingress node. However, this policy could trigger the problem of blocking since the control packet might not be able to reserve the channel successfully, but the sender still sent the data burst in the bufferless OBS network. In such a case, the router would simply drop those data bursts since no available channels were reserved for them. This behavior is defined as blocking. The occurrence of blocking acts as the measurement of network performance [23,24].

In an OPS network, each packet contains both the control header and the data. As the header and the data coexist in the packet, the router processes them separately. For the incoming packets, the router analyzes the header optically, namely, all-optical packet switching [25]. The routing algorithm determines the output fiber and wavelength to conduct the forwarding. However, for a bufferless OPS network, we consider the case that its output ports have no buffers and the packet loss might occur when all wavelengths in the output fiber are occupied [26].

### B. EFPA for Bufferless OBS/OPS Networks

Apart from the discrete event simulation, the EFPA is also a popular method for estimating the blocking probability in OBS/OPS networks. Rosberg first applied the EFPA to optical burst switched networks in 2003 [11]. After that, an advanced algorithm based on EFPA was proposed for OBS networks with different routing strategies, protections, and contention resolution methods [12]. In general, the idea of EFPA is to decouple the network into independent links and to assume the traffic is a collection of Poisson processes. The EFPA is required to go through a number of iterations until convergence is achieved. In each iteration, the blocking probability of each link is recalculated and fed to the next iteration. The EFPA can calculate the network blocking probability in an efficient manner due to its simple model structure. However, the assumptions of independent links and the Poisson process may introduce significant errors. For instance, the blocking probability could be overestimated by EFPA under low traffic conditions. A detailed performance analysis of EFPA is provided in Section V.

### C. Neural Network and Networking

A neural network is a popular tool for solving networking issues. For instance, a neural network approach was proposed for TCP congestion control [27]. This approach considered the backpropagation, a commonly used neural network approach, for learning the active queue management (AQM) parameters. After that, the trained neural network acted as a controller to adjust the window size autonomously. In Ref. [28], an extension work for the *ad hoc* wireless network was proposed. Apart from TCP congestion control, a neural network could also estimate the blocking probability. In Ref. [13], the multilayer perceptron (MLP) model was used to estimate blocking probability. This MLP approach is very efficient in terms of evaluation speed.

### D. ELM

For a function approximation problem, we denote the training set as $\mathbb{D}_t = \{(\boldsymbol{x}_k, p_k): \boldsymbol{x}_k \in \mathbb{R}^d, p_k \in \mathbb{R}, k = 1, \ldots, N\}$, where $\boldsymbol{x}_k$ and $p_k$ are the input vector and the target output of the $k$th sample, respectively. In the SLFN approach [14–16], the network output is given by

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

where $g_i(\boldsymbol{x})$ is the output of the $i$th hidden node, and $\beta_i$ is the connection weight between the $i$th hidden node and the output node. For the hidden node in an SLFN, we can choose many kinds of activation functions, such as sigmoid functions [29] and radial basis functions [30]. Since many articles [29,31,32] recommended the sigmoid function as the activation function for the ELM-based SLFN, this paper considers the sigmoid function as the activation function for the hidden nodes.

With the sigmoid functions, the output of the $i$th hidden node is given by

$$g_i(\boldsymbol{x}) = \frac{1}{1+\exp\{-(\boldsymbol{a}_i^T\boldsymbol{x}+b_i)\}}, \tag{2}$$

where $b_i$ is the bias term of the $i$th hidden node, and $\boldsymbol{a}_i$ is the input weight vector of the $i$th hidden node.

For the ELM approach [17,18], the bias terms' $b_i$'s and the input weights' $\boldsymbol{a}_i$'s are generated randomly. We only need to adjust the output weights' $\beta_i$'s. When we use $n$ hidden nodes for approximation, the MSE is given by

$$\mathcal{E} = \sum_{k=1}^{N}\left(p_k - \sum_{i=1}^{n}\beta_i g_i(\boldsymbol{x}_k)\right)^2 = \left\|\boldsymbol{p} - \sum_{i=1}^{n}\beta_i \boldsymbol{g}_i\right\|_2^2, \tag{3}$$

where $\boldsymbol{p} = [p_1, \ldots, p_N]^T$, and $\boldsymbol{g}_i = [g_i(\boldsymbol{x}_1), \ldots, g_i(\boldsymbol{x}_N)]^T$.

For the I-ELM algorithm [17], at the $n$th step, we add a new node $g_n(\cdot)$ into the network. We need to determine the output weight $\beta_n$ of the newly additive node, but we do not change all the previously trained weights' $\beta_i$'s, for $i = 1, \ldots, n-1$. Recall that at the $n$th step, the training error is given by

$$\mathcal{E}_n = \left\|\boldsymbol{p} - \sum_{i=1}^{n}\beta_i^2 \boldsymbol{g}_i\right\|_2^2 = \|\boldsymbol{e}_{n-1} - \beta_n \boldsymbol{g}_n\|_2^2, \tag{4}$$

where $\boldsymbol{e}_{n-1} = \boldsymbol{p} - \sum_{i=1}^{n-1}\beta_i\boldsymbol{g}_i$. Hence, to minimize $\mathcal{E}_n$ (without modifying the previous weights), the output weight $\beta_n$ is given by

$$\beta_n = \frac{\boldsymbol{e}_{n-1}^T\boldsymbol{g}_n}{\|\boldsymbol{g}_n\|_2^2}. \tag{5}$$

Algorithm 1 summarizes the steps in the I-ELM. We can easily observe that the complexity for each iteration is $O(N)$ only.

**Algorithm 1:** I-ELM

```
1: Initialization: Let the number of hidden nodes $n = 0$ and the residual error $e_0 = p$.
2: Define the termination condition $\varepsilon$.
3: while $n \le n_{\max}$ and $(\mathcal{E}_{n-1} - \mathcal{E}_n)/\mathcal{E}_{n-1} > \varepsilon$ do
4:     $n = n + 1$.
5:     Add a new hidden node $g_n(\cdot)$ to the network, where $(a_n, b_n)$ are randomly generated.
6:     Compute the new weight $\beta_n$: $\beta_n = \frac{e_{n-1}^T g_n}{\|g_n\|_2^2}$.
7:     $e_n = e_{n-1} - \beta_n g_n$.
8: end while
```

## E. ADMM

The ADMM framework [20,21] is an iterative approach for solving convex problems. It considers the following constraint optimization problem:

$$\min_{\boldsymbol{\beta},\boldsymbol{u}} : \psi(\boldsymbol{\beta}) + \phi(\boldsymbol{u}) \text{s.t. } \boldsymbol{C\beta} + \boldsymbol{Du} = \boldsymbol{v}, \tag{6}$$

where $\psi$ and $\phi$ are the cost terms in the objective function, and $\boldsymbol{\beta}$ and $\boldsymbol{u}$ are decision variable vectors in the optimization problem.

In the ADMM framework, we first construct an augmented Lagrangian given by

$$L(\boldsymbol{\beta},\boldsymbol{u},\boldsymbol{\gamma}) = \psi(\boldsymbol{\beta}) + \phi(\boldsymbol{u}) + \boldsymbol{\gamma}^T(\boldsymbol{C\beta}+\boldsymbol{Du}-\boldsymbol{v}) + \frac{\rho}{2}\|\boldsymbol{C\beta}+\boldsymbol{Du}-\boldsymbol{v}\|_2^2, \tag{7}$$

where $\boldsymbol{\gamma}$ is the Lagrangian vector, and $\rho$ is a positive penalty parameter. The iterative scheme for $\{\boldsymbol{\beta},\boldsymbol{u},\boldsymbol{\gamma}\}$ is given by

$$\boldsymbol{\beta}^{k+1} = \arg\min_{\boldsymbol{\beta}} L(\boldsymbol{\beta},\boldsymbol{u}^k,\boldsymbol{\gamma}^k), \tag{8a}$$

$$\boldsymbol{u}^{k+1} = \arg\min_{\boldsymbol{u}} L(\boldsymbol{\beta}^{k+1},\boldsymbol{u},\boldsymbol{\gamma}^k), \tag{8b}$$

$$\boldsymbol{\gamma}^{k+1} = \boldsymbol{\gamma}^k + \rho(\boldsymbol{C\beta}^{k+1}+\boldsymbol{Du}^{k+1}-\boldsymbol{v}). \tag{8c}$$

## III. Details of the Proposed Training Models

### A. Methodology

Given an OBS/OPS network and some network conditions $\{\boldsymbol{x}_k : \boldsymbol{x}_k \in \mathbb{R}^d\}$, we can use a computer simulation to obtain the corresponding blocking probability, $\{p_k : p_k \in \mathbb{R}_{>0}\}$. Here, we assume that there are $d$ network parameters that describe the properties/conditions of the OBS/OPS network; and we would like to train an SLFN to learn the mapping from the network conditions to the blocking probability.

Since the dynamic range of blocking probability was very wide, the percentage error of the approximation for small blocking probability values may be very large when the blocking probability values obtained by simulation are directly used as the target outputs of the SLFN. Therefore, the I-ELM algorithm, which trains the SLFN in the original domain, has a similar problem as in the Araujo's approach [13].

To handle this problem, we train our SLFN to learn the log version of the blocking probability, i.e., $\{\tilde{p}_k = \log p_k\}$, instead. In this regard, the training set is given by

$$\tilde{\mathbb{D}}_t = \{(\boldsymbol{x}_k, \tilde{p}_k = \log p_k) : \boldsymbol{x}_k \in \mathbb{R}^d, \tilde{p}_k \in \mathbb{R}_{<0}, k = 1, \ldots, N\}. \tag{9}$$

Our training objective will then become

$$\tilde{\mathcal{E}} = \sum_{k=1}^{N}\left(\tilde{p}_k - \sum_{i=1}^{n}\beta_i g_i(\boldsymbol{x}_k)\right)^2 = \left\|\tilde{\boldsymbol{p}} - \sum_{i=1}^{n}\beta_i \boldsymbol{g}_i\right\|_2^2, \tag{10}$$

where $\tilde{\boldsymbol{p}} = [\tilde{p}_1, \ldots, \tilde{p}_N]^T$.

Using the I-ELM concept, we obtain a novel incremental ELM algorithm, namely, the Log-I-ELM algorithm, for minimizing the objective function $\tilde{\mathcal{E}}$. Algorithm 2 summarizes the key steps of the proposed algorithm. The main difference between the I-ELM algorithm and the Log-I-ELM algorithm is that the Log-I-ELM learns the mapping from the network conditions to the log blocking probability.

**Algorithm 2:** Log-I-ELM

```
1: Initialization: Let the number of hidden nodes $n = 0$ and the residual error $\boldsymbol{e}_0 = \tilde{\boldsymbol{p}}$.
2: Define the termination condition $\varepsilon$.
3: while $n \le n_{\max}$ and $(\tilde{\mathcal{E}}_{n-1} - \tilde{\mathcal{E}}_n)/\tilde{\mathcal{E}}_{n-1} > \varepsilon$ do
4:     $n = n + 1$.
5:     Add a new hidden node $g_n(\cdot)$ to the network, where $(\boldsymbol{a}_n, b_n)$ are randomly generated.
6:     Compute the new weight $\beta_n$: $\beta_n = \frac{\boldsymbol{e}_{n-1}^T \boldsymbol{g}_n}{\|\boldsymbol{g}_n\|_2^2}$.
7: $\boldsymbol{e}_n = \boldsymbol{e}_{n-1} - \beta_n \boldsymbol{g}_n$.
8: end while
```

After the Log-I-ELM process, we obtain $n_e$ hidden nodes. We describe these nodes by $\{g_1(\boldsymbol{x}), \ldots, g_{n_e}(\boldsymbol{x})\}$. Some of them may be unimportant. Here, we propose to use the ADMM approach to remove some unimportant nodes.

With the ADMM framework, there are two algorithms. One is called ADMM-I-ELM. The ADMM-I-ELM trains the SLFN first and then uses the ADMM framework to remove some unimportant hidden nodes. Another one is ADMM-Log-I-ELM. The ADMM-Log-I-ELM trains the SLFN first and then uses the ADMM framework to remove some unimportant hidden nodes.

With the $n_e$ hidden nodes, the approximation error is given by

$$\tilde{\mathcal{E}} = \left\|\tilde{\boldsymbol{p}} - \sum_{i=1}^{n_e}\beta_i \boldsymbol{g}_i\right\|_2^2, \tag{11}$$

where $\tilde{\boldsymbol{p}} = [\tilde{p}_1, \ldots, \tilde{p}_N]^T$. If we are allowed to retrain the network, we can use the standard least squares approach to find the output weights' $\beta_i$'s. However, the standard least squares approach is unable to remove the unimportant nodes.

In order to delete some unimportant nodes, we can add an $\ell_1$ norm penalty term into the objective function. With the penalty term, some output weights' $\beta_i$'s are dragged to zero, and we remove their corresponding nodes after the training. The modified objective function is then given by

$$\tilde{\mathcal{J}} = \frac{1}{2}\|\tilde{\boldsymbol{p}} - \boldsymbol{\Phi}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1, \tag{12}$$

$$\Phi = \begin{pmatrix} g_1(\boldsymbol{x}_1) & \cdots & \cdots & g_{n_e}(\boldsymbol{x}_1) \\ g_1(\boldsymbol{x}_2) & \ddots & \cdots & g_{n_e}(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ g_1(\boldsymbol{x}_N) & \cdots & \cdots & g_{n_e}(\boldsymbol{x}_N) \end{pmatrix}, \tag{13}$$

where $\lambda$ is a trade-off parameter between the approximation error of the log blocking probability and the number of the selected nodes. We can adjust $\lambda$ to balance between the number of hidden nodes and the approximation ability. When a large $\lambda$ is used, we would like to have an SLFN with less hidden nodes, while when a small $\lambda$ is used, we would like to have an SLFN with more approximation ability.

The unconstraint optimization in Eq. (12) can be solved by the ADMM framework. We first introduce a dummy variable $\boldsymbol{u}$. Then, the unconstraint optimization problem becomes a constraint optimization problem given by

$$\min_{\boldsymbol{\beta},\boldsymbol{u}} \frac{1}{2}\|\tilde{\boldsymbol{p}} - \Phi\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{u}\|_1, \text{s.t.} : \boldsymbol{u} = \boldsymbol{\beta}. \tag{14}$$

The augmented Lagrangian of Eq. (14) is given by

$$\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{u}, \boldsymbol{\gamma}) = \frac{1}{2}\|\tilde{\boldsymbol{p}} - \boldsymbol{\Phi}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{u}\|_1 + \boldsymbol{\gamma}^T(\boldsymbol{u} - \boldsymbol{\beta}) + \frac{\rho}{2}\|\boldsymbol{u} - \boldsymbol{\beta}\|_2^2. \tag{15}$$

Therefore, the corresponding ADMM updates are given as Eqs. (17), (19) and (21).

**Computing $\boldsymbol{\beta}^{k+1}$:**

By fixing $\boldsymbol{u}$ as $\boldsymbol{u}^k$ and $\boldsymbol{\gamma}$ as $\boldsymbol{\gamma}^k$ in Eq. (15), we can update $\boldsymbol{\beta}^{k+1}$ by

$$\begin{aligned}\boldsymbol{\beta}^{k+1} &= \arg\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{u}^k, \boldsymbol{\gamma}^k) \\ &= \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\tilde{\boldsymbol{p}} - \Phi\boldsymbol{\beta}\|_2^2 + \boldsymbol{\gamma}^{kT}(\boldsymbol{u}^k - \beta) + \frac{\rho}{2}\|\boldsymbol{u}^k - \beta\|_2^2 \\ &= \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\tilde{\boldsymbol{p}} - \Phi\boldsymbol{\beta}\|_2^2 + \frac{\rho}{2}\left\|\boldsymbol{u}^k - \boldsymbol{\beta} + \frac{1}{\rho}\boldsymbol{\gamma}^k\right\|_2^2.\end{aligned} \tag{16}$$

From Eq. (16), we obtain

$$\boldsymbol{\beta}^{k+1} = [\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \rho\boldsymbol{I}]^{-1}[\boldsymbol{\Phi}^T\tilde{\boldsymbol{p}} + \rho\boldsymbol{u}^k + \gamma^k], \tag{17}$$

where $\boldsymbol{I}$ is an identity matrix.

**Computing $\boldsymbol{u}^{k+1}$:**

By fixing $\boldsymbol{\beta}$ as $\boldsymbol{\beta}^{k+1}$ and $\boldsymbol{\gamma}$ as $\boldsymbol{\gamma}^k$ in Eq. (15), we can update $\boldsymbol{u}^{k+1}$ by

$$\boldsymbol{u}^{k+1} = \arg\min_{\boldsymbol{u}} \lambda\|\boldsymbol{u}\|_1 + \frac{\rho}{2}\left\|\boldsymbol{u} - \boldsymbol{\beta}^{k+1} + \frac{1}{\rho}\boldsymbol{\gamma}^k\right\|_2^2. \tag{18}$$

Although the term $\lambda\|\boldsymbol{u}\|_1$ in Eq. (18) is nondifferentiable, we can obtain the closed-form solution by using subdifferential calculus [21] given by

$$\boldsymbol{u}^{k+1} = \mathcal{S}_{\rho/\lambda}\left(\boldsymbol{\beta}^{k+1} - \frac{1}{\rho}\boldsymbol{\gamma}^{k+1}\right), \tag{19}$$

where $\mathcal{S}_\zeta(\cdot)$ is a soft-threshold operator [21] given by

$$\mathcal{S}_\zeta(\theta) = \mathrm{sgn}(\theta) \cdot (|\theta| - \zeta). \tag{20}$$

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

**Computing** $\gamma^{k+1}$:

$$\gamma^{k+1} = \gamma^k + \rho(\boldsymbol{u}^{k+1} - \boldsymbol{\beta}^{k+1}). \tag{21}$$

## *B. Simulation Parameters*

We choose six input parameters of an optical network, shown in Table I (namely, $E$, $D$, $C$, $W$, APL, and CR), as the neural network inputs. These parameters are much related to the properties of network input traffic (i.e., $E$ and $D$), optical links (i.e., $C$ and $W$), and network topology (i.e., APL and CR). These six chosen parameters are the independent input parameters for the network simulation. They define the major properties and configurations of a network. Four of the parameters $E$, $W$, APL, and CR are independent input variables inherited from [13]. Since our focus is on the network layer rather than the physical layer, the other parameters chosen in Ref. [13] are neglected in our model. Apart from these four parameters, we consider two more independent parameters $D$ and $C$. $D$ (the difference between maximum and minimum traffic load per S-D pair) is used to complement $E$ (the mean traffic load per S-D pair) for describing the input traffic properties. $C$ (the number of channels per wavelength) is used to complement $W$ (the number of wavelengths) for describing the optical link properties.

The definitions of the six parameters are as follows. First, the mean traffic load per S-D pair ($E$) is the major factor to determine the blocking probability. $E$ is measured in Erlangs and means the offered traffic between the source and destination nodes. In general, a data burst has a high chance of being blocked in a heavy traffic network [33]. Therefore, the blocking probability would increase with $E$.

Second, we represent the difference between the maximum and the minimum offered load per S-D pair by the parameter $D$, which represents the variation of the network traffic. The offered load to each S-D pair is assumed to be uniformly distributed in between the interval $[E - D/2, E + D/2]$. If we fix the value of $E$ and increase $D$, the offered load will have a wider range, and therefore, the S-D pair traffic will be more bursty. Then, the number of blocked bursts/packets of some links would be extremely large, which would lead to a network with larger blocking probability.

TABLE I
INDEPENDENT INPUT PARAMETERS IN THE SIMULATION

| Parameter | Definition |
|---|---|
| $E$ | Mean traffic load in S-D pair (Erlangs) |
| $D$ | Difference between max and min traffic load in S-D pair |
| $C$ | Number of channels per wavelength |
| $W$ | Number of wavelengths |
| APL | Average path length |
| CR | Concentration of route |

Third and fourth, $C$ and $W$ are the number of channels per wavelength and the number of wavelengths in each link, respectively. Suppose that a link is composed of $F$ fibers, each fiber supports $W$ wavelengths, and each wavelength supports $S$ subwavelength channels (e.g., TDM). For the case of no wavelength conversion, a new burst randomly selects a free channel for transmission and uses the same wavelength for its entire travel. The number of channels per wavelength $C$ would then be the number of fibers $F$ times the number of subwavelength channels $S$, i.e., $C = F \times S$. When the traffic to each S-D pair and $C$ are fixed, increasing $W$ will mean that each wavelength had a lighter load. Then, the blocking probability will decrease. When the traffic to each S-D pair and $W$ are fixed, increasing $C$ will reduce the network blocking probability.

Fifth, the average path length APL represents the mean distance between the S-D pairs. That is also the average number of hops that the traffic would pass through. The value is calculated based on the routing table. Lowering the APL value would lower the blocking probability [34].

Sixth, the concentration of routes CR, which is defined by [13], states the average number of routes that occur in the optical link. It is an indicator to reveal the load balancing of the network. Meanwhile, the value of CR is calculated by their proposed algorithm. Increasing the CR value would also increase the blocking probability.

These six input parameters have a profound influence on our training model. For $E$, $W$, APL, and CR, the work in Ref. [13] had analyzed their influence. They introduced a parameter, the traffic load $E$, to properly assess the impact on the performance of the blocking probability estimation. They presented a correlation matrix for choosing the appropriate set of variables, and they found that $W$ had the lowest correlation among their set of variables, i.e., the most important variables. They also chose their parameters based on the MSE changes with respect to the changes of the number of variables and identified that APL and CR were significant to the training model. Based on their justification, we inherited these four parameters as the input of our neural network model.

With similar approaches, based on the MSE changes with respect to the changes of the number of variables, we identified two more variables $D$ and $C$, which provide more information about the properties of network traffic and the optical link. Table II shows that the MSE values of including $D$ and $C$ had a considerable reduction, which were decreased from $1.649 \times 10^{-3}$ to $1.243 \times 10^{-3}$ and $9.010 \times 10^{-4}$, respectively. Using the formulation in Ref. [13], the MSE values were decreased 0.25 and 0.45 by proportion, and both were larger than their suggested threshold 0.05. In addition, Table II shows that including both $D$ and $C$ could further decrease the MSE value to $4.731 \times 10^{-4}$. Therefore, these six input variables were

TABLE II
Trained Models With $P$ Parameters

| $P$ | Input Parameters | MSE |
|---|---|---|
| 4 | $E$, $W$, CR, APL | $1.649 \times 10^{-3}$ |
| 5 | $E$, $W$, CR, APL, $D$ | $1.243 \times 10^{-3}$ |
| 5 | $E$, $W$, CR, APL, $C$ | $9.010 \times 10^{-4}$ |
| 6 | $E$, $W$, CR, APL, $D$, $C$ | $4.731 \times 10^{-4}$ |

crucial and hence used in our ELM approach to accurately evaluate the blocking probability.

## IV. Experimental Setup

### A. Bufferless OBS/OPS Network Simulations

For the bufferless OBS/OPS network simulation, we considered the NSFNet topology. In Fig. 1, 13 nodes were present and 16 optical links were installed among the nodes. There were a total of 78 S-D pairs in the network. The pairs were bidirectional, i.e., the nodes could communicate with each other.

In the simulation, the Markov-chain simulation model was implemented. The arrival process was assumed to be Poisson, and the service time was assumed to be exponentially distributed. As mentioned in Section III, we use six parameters for the simulation and our training model. Since our interested range of blocking probability was $[10^{-5}–10^{-1}]$, we adjusted these parameters to generate a dataset corresponding to the blocking probability within our interested range.

The generated dataset was then used for our training model. The computational time for the ADMM-I-ELM and ADMM-Log-I-ELM required 73 and 102 min, respectively. Like the existing neural network approach, the evaluation of the estimated blocking probability took less than a second to complete.

The specific ranges of the six parameters were chosen based on the following two reasons. First, Araujo *et al.* in Ref. [13] suggested that the traffic load should be [60–230] Erlangs for the whole network, and we adopted this setting. However, instead of measuring the traffic load of the whole network, we measured the mean of the traffic load per S-D pair $E$, which simplified the equations. It was noticed that [60–230] Erlangs of a 78 S-D pair network would be [0.8–3.0] Erlangs per S-D pair. The configuration of the number of wavelengths $W$ was usually [1–10] and the whole capacity of the links was usually in the range of [80–120], i.e., capacity equal to $W$ times $C$ [35,36]. We directly adopted [1, 10] as the range of our number of wavelengths $W$. With the selected range of $W$ [1–10] and the capacity [80–120], the number of channels per wavelength $C$ was obtained through the capacity divided by $W$, which could be in between 8 and 120.

Second, the difference between the maximum and minimum traffic load per S-D pair $D$ was [0.0–1.0], the average path length APL was in the range of [2.52, 4.62], and the concentration of routes CR was in the range of [1.81–4.81]. These were just some empirical settings, such that the generated dataset would have their blocking probability within $[10^{-5}–10^{-1}]$.

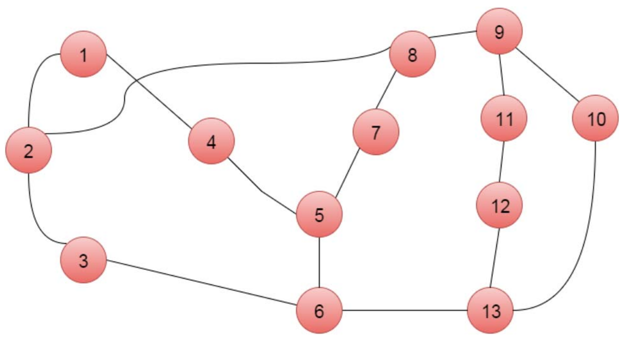

Fig. 1. Thirteen-node NSFNet topology.

### B. Dataset Prepossessing and Settings

Before the training stage, the dataset would require some prepossessing. To start with, we defined an attribute to be all the values in the dataset of an individual parameter. Each attribute was normalized to prevent the occurrence of outliers. The normalization was carried out with the attribute mean and the attribute standard deviation. Then, the dataset was separated into three classes based on the intervals of blocking probability. Table III shows the classification criterion. With this criterion, each class was guaranteed to contain at least 6000 entries for our dataset. For the training set, 10,000 entries were selected randomly from the three classes. Each class provided one-third of the training set entries. For the test set, 10,000 entries were selected with the same approach. The classification design was intended to balance the blocking probability of the selected entries.

During the training stage, the training set was learned by I-ELM and Log-I-ELM as mentioned in Section III, and two sets of hidden nodes were generated. Then, we removed the unimportant nodes by the ADMM optimization. In the ADMM empirical setting, the parameter $\rho$ was set to 2 and the trade-off parameter $\lambda$ was in the range of $[10^{-4}–10^{0}]$. In I-ELM and Log-I-ELM, we set a termination condition to stop the insertion of hidden nodes. Our termination condition was based on the change in the residue error between two consecutive iterations. In Algorithms 1 and 2, when

$$\frac{\|e_{n-1}\|_2^2 - \|e_n\|_2^2}{\|e_{n-1}\|_2^2} < 0.01,$$

TABLE III
Classes of Datasets

| Class | Lower Bound | Upper Bound |
|---|---|---|
| 1 | 1E-5 | 1E-3 |
| 2 | 1E-3 | 1E-1 |
| 3 | 1E-1 | 0.25 |

we would stop the ELM training. To measure the performance of the two algorithms, we considered two metrics. One was the MSE of the probability, which was given by

$$\mathrm{MSE} = \frac{1}{N}\sum_{k=1}^{N}(p_k - \hat{p}_k)^2, \tag{22}$$

where $p_k$'s were the true probability values, and $\hat{p}_k$'s were the estimated probability values. It should be noted that the MSE value was not a good measurement for the approximation of probability values. For blocking probability, the dynamic range was very wide. Small MSE values did not mean that we had a good approximation for small blocking probability values.

The other one was relative error given by

$$\text{relative error} = \frac{1}{N}\sum_{k=1}^{N}\frac{|p_k - \hat{p}_k|}{p_k}. \tag{23}$$

It should be noted that the relative error gave out a better representation for the approximation error for the high dynamic range data.

## V. Results

### A. Comparison Between I-ELM and Log-I-ELM

The analysis of results generated in different stages is presented in the following subsections.

In the first part of the experiment, we used the two algorithms, I-ELM and Log-I-ELM, to build two SLFNs for the estimation of blocking probability. For the I-ELM case, the training was terminated after 2791 hidden nodes being inserted into the network. For the Log-I-ELM case, the training was terminated after 4591 hidden nodes being inserted into the network.

Figure 2 presents the test set MSE values of estimated blocking probability versus the number of hidden nodes. From Fig. 2(a), for the I-ELM case, the MSE value was equal to 0.001122, when 2791 hidden nodes were used. From Fig. 2(b), for the Log-I-ELM case, the MSE value was equal to 0.0306 when 4591 hidden nodes were used. Comparing the test set MSE values, the test set MSE of Log-I-ELM was nearly 10 times larger than that of I-ELM. This phenomenon should be related to their training domain. For the I-ELM case, the training objective was

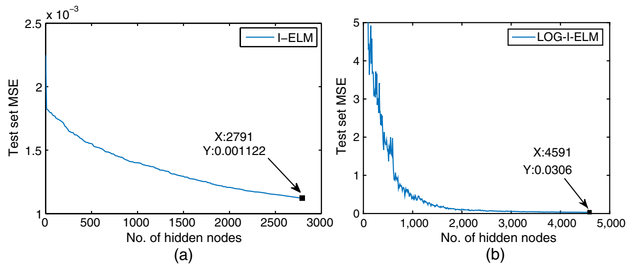

Fig. 2. Test set MSE versus number of hidden nodes of I-ELM and Log-I-ELM.

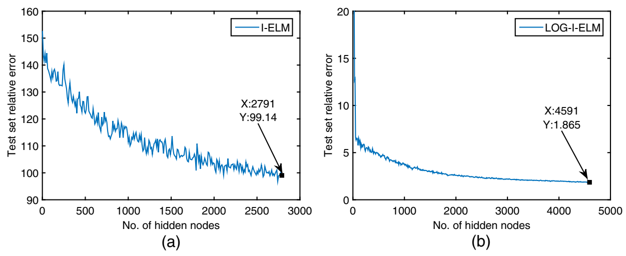

Fig. 3. Test set relative error versus number of hidden nodes of I-ELM and Log-I-ELM.

the MSE of blocking probability, whose dynamic range was from $10^{-5}$ to $10^{-1}$. Hence, although the overall MSE of the I-ELM is small, it does not mean that the I-ELM can have a good approximation for the small blocking probability range, and we will demonstrate this in Section V.C.

Figures 3(a) and 3(b) present the test set relative errors of the two approaches. Both test set relative errors were gradually decreasing with respect to the number of hidden nodes. In Fig. 3(a), the relative error was equal to 99.14. This was much larger than that of Log-I-ELM, which was 1.865. Therefore, the blocking probability estimated by the Log-I-ELM was more accurate than the one done by the I-ELM over the data range.[1] Although the test set MSE of the I-ELM is far less than the one of the Log-I-ELM, the relative error shows that the Log-I-ELM algorithm is better than the I-ELM algorithm. This explains why our focus should be on the order of magnitude (relative error) rather than the MSE. With these results, we find that training for the estimation of blocking probability should be processed in the log domain.

From the two figures, the number of used hidden nodes in both I-ELM and Log-I-ELM were still large. Thus, the two trained SLFNs should be further optimized by the ADMM framework so as to remove some unimportant nodes.

### B. Comparison Between ADMM-I-ELM and ADMM-Log-I-ELM

The ADMM framework was used for optimizing our trained models. In the ADMM framework, by varying the weighting factor $\lambda$, we could control the number of hidden nodes used. Figure 4 shows the test set MSE versus the number of hidden nodes. The figure shows that using the ADMM framework could reduce the approximation error and the number of hidden nodes used. For instance, in the ADMM-I-ELM case, we only needed 1427 hidden nodes to achieve a test set MSE value 0.0001875. For the ADMM-Log-I-ELM case, we only needed 2968 hidden nodes to achieve a test set MSE value 0.0004732. Again, the test set MSE value in the ADMM-I-ELM is less than the one in the ADMM-Log-I-ELM. It is not surprising that this

[1]We will demonstrate this in Section V.C and Figs. 6 and 7.

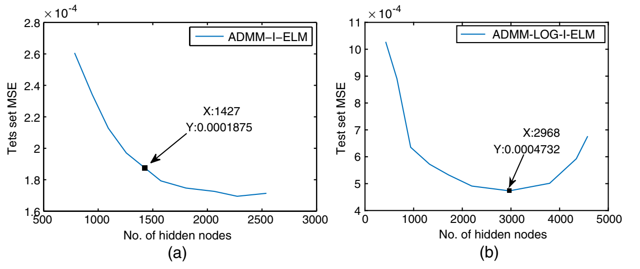

Fig. 4. MSE versus the number of hidden nodes of ADMM-I-ELM and ADMM-Log-I-ELM.

phenomenon happens again. Therefore, in our case, the MSE only tells us the validity of the models, but it is not a guideline to choose the trained output weights among the models.

In order to select the hidden nodes to generate the most accurate output values, the relative error in both models should be observed. In Fig. 5(a), the ADMM-I-ELM algorithm used 1427 nodes to obtain the minimum relative error with value 19.66. Thus, this set of hidden nodes was considered to represent the ADMM-I-ELM model. Compared with the I-ELM, the ADMM-I-ELM reduced the number of hidden nodes from 2791 to 1427. Also, the test set relative error declined from 99.14 to 19.66.

In Fig. 5(b), the ADMM-Log-I-ELM algorithm used 2968 nodes to obtain a minimum relative error with value 0.2903. Through the ADMM optimization, the relative error dropped from 1.81 to 0.29, and the required number of hidden nodes decreased from 4591 to 2968.

To sum up, in terms of relative error and the required number of hidden nodes, the ADMM framework can significantly improve the performance and the resource requirement. In addition, the relative error of the ADMM-Log-I-ELM is much better than that of the ADMM-I-ELM. That means, the ADMM-Log-I-ELM algorithm is much more suitable for the estimation of blocking probability, which is in the high dynamic range.

## *C. Estimation of Blocking Probability by ELM*

With the ADMM framework, we could construct the SLFNs and use them to estimate the blocking probability. In order to test the accuracy, the dataset was chosen to

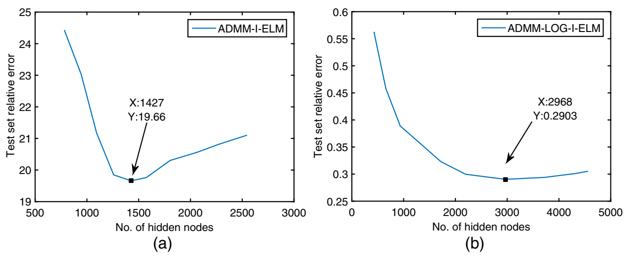

Fig. 5. Relative error versus number of hidden nodes of ADMM-I-ELM and ADMM-Log-I-ELM.

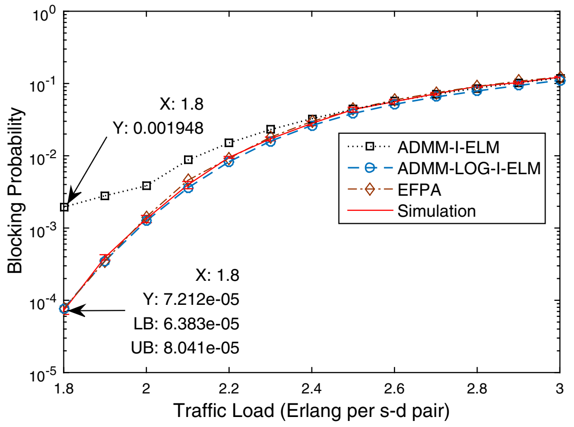

Fig. 6. Blocking probability estimation by ADMM-I-ELM, ADMM-Log-I-ELM, and EFPA with setting $D = 0.2$, $C = 120$, $W = 1$, APL $= 3.08$, and CR $= 1.8125$.

have a wide range of blocking probability, and the target range was $[10^{-5}–10^{-1}]$.

Figure 6 demonstrates the performance of the estimated blocking probability versus the traffic load $E$ of the OBS networks. In this setting, we treated the other five inputs as constant, where $D = 0.2$, $C = 120$, $W = 1$, APL $= 3.08$, and CR $= 1.8125$. Meanwhile, $E$ was in the range of [1.8–3]. In Fig. 6, we notice that our proposed approaches both fell within the 95% confidence interval of the simulation when $E$ was above 2.4 Erlangs per pair. In other words, for the ADMM-I-ELM and ADMM-Log-I-ELM algorithms, the estimation was accurate in the range of blocking probability $[10^{-2}–10^{-1}]$. However, the ADMM-I-ELM seemed to overestimate the value, while $E$ was less than 2.4 Erlangs per pair. When $E$ was 1.8 Erlangs per pair, the estimated value was nearly 25 times higher than that of the simulation. It reveals that this algorithm works poorly in the low blocking probability region.

On the other hand, the ADMM-Log-I-ELM algorithm could approximate the blocking probability value accurately in the low blocking probability region. For example, when $E$ was 1.8 Erlangs per pair, the estimated blocking probability of $7.855 \times 10^{-5}$ was within the 95% confidence interval of the simulations, i.e., $[6.383 \times 10^{-5}–8.041 \times 10^{-5}]$. Under this setting, the ADMM-Log-I-ELM performs excellently for the whole range of blocking probability.

A similar observation can be made in Fig. 7. When $E$ was 0.9 Erlang per pair, the estimated value of ADMM-I-ELM was nearly 29 times higher than that of the simulation. For the ADMM-Log-I-ELM case, the estimated value was $1.534 \times 10^{-5}$, which was within the 95% confidence interval of the simulation, i.e., $[1.406 \times 10^{-5}–1.759 \times 10^{-5}]$. Generally, the estimated values of ADMM-Log-I-ELM are within the 95% confidence interval of simulation, whereas the ADMM-I-ELM tends to overestimate by a factor of more than 20 times.

Based on these results, we observe that the ADMM-Log-I-ELM algorithm is better than ADMM-I-ELM among different situations. It is believed that the training domain

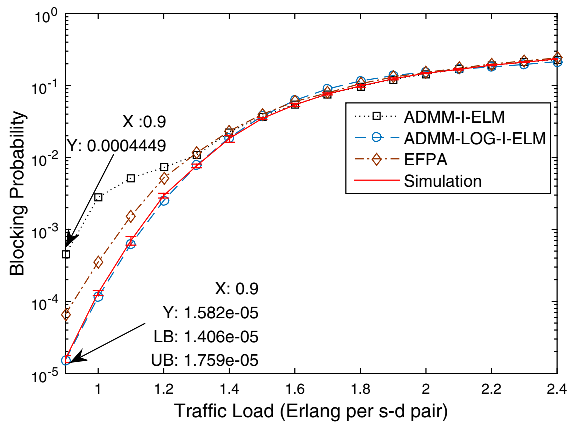

Fig. 7. Blocking probability estimation by ADMM-I-ELM, ADMM-Log-I-ELM, and EFPA with setting $D = 0.2$, $C = 40$, $W = 2$, APL = 3.08, and CR = 1.8125.

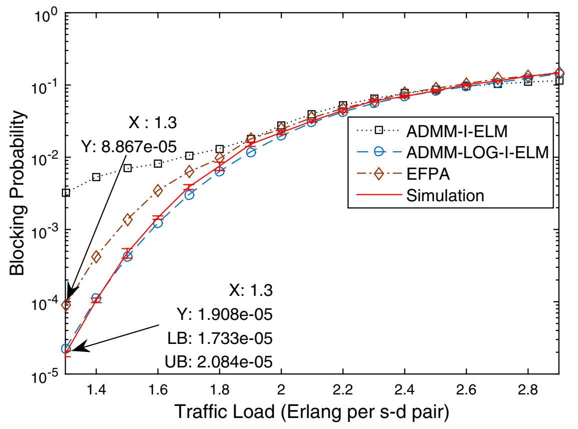

Fig. 9. Blocking probability estimation by ADMM-I-ELM, ADMM-Log-I-ELM, and EFPA with setting $D = 0.4$, $C = 50$, $W = 2$, APL = 2.628, and CR = 2.25.

is the major reason to explain this problem. As expected, training in the log domain avoided overfocusing large output values, such that the small values have a chance to gain the attention of the hidden nodes. In this manner, the algorithm estimates the blocking probability accurately for the values in the low region.

## D. Comparison With EFPA

We also compared our proposed ADMM-Log-I-ELM algorithm with the EFPA model. Figure 6 shows the results generated by EFPA. It was similar to the one by simulation and ADMM-Log-I-ELM when the number of wavelengths $W$ was 1. However, the EFPA overestimated the blocking probability by nearly 3–5 times when the value of $W$ was larger than 1, as shown in Figs. 7–10. In particular, as shown in Fig. 8, when $E$ was 1 Erlang per pair and $W$ was 3, the ADMM-Log-I-ELM estimation $3.837 \times 10^{-5}$ was in the 95% confidence interval of the simulation. On the other hand, the EFPA estimation was $2.452 \times 10^{-4}$, which was nearly 5 times larger than the simulated value of $4.749 \times 10^{-5}$. This phenomenon also happened in Figs. 9 and 10 for small blocking regions.

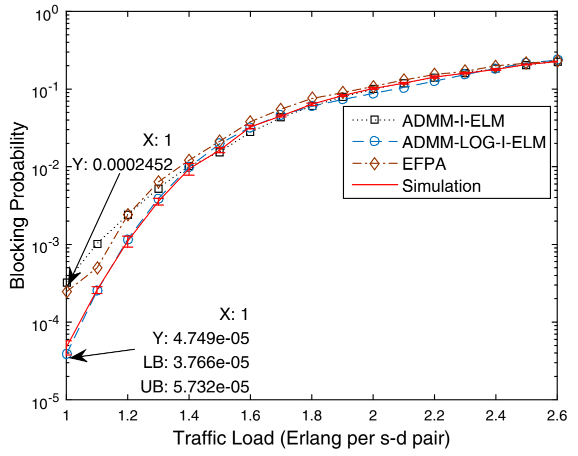

Fig. 8. Blocking probability estimation by ADMM-I-ELM, ADMM-Log-I-ELM, and EFPA with setting $D = 0.4$, $C = 40$, $W = 3$, APL = 3.75, and CR = 3.

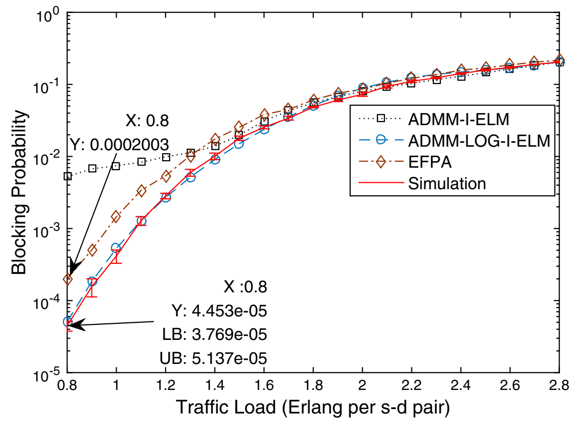

Fig. 10. Blocking probability estimation by ADMM-I-ELM, ADMM-Log-I-ELM, and EFPA with setting $D = 0.8$, $C = 18$, $W = 5$, APL = 2.628, and CR = 2.25.

For an OBS network, a newly arrived burst uniformly and randomly chooses a free channel among all available channels. However, in EFPA, we assume that a newly arrived burst randomly chooses a wavelength first instead. When all the channels on this wavelength are busy, the new burst is deflected to other wavelengths until it finally finds a free channel. Furthermore, when all the channels in all the wavelengths are busy, it would be blocked. Therefore, EFPA overestimates the network blocking probability.

## VI. Conclusion

This paper proposed two ELM models, namely, ADMM-I-ELM and ADMM-Log-I-ELM, to estimate the blocking probability in the bufferless OBS/OPS network. For the ADMM-I-ELM algorithm, the incremental learning approach is conducted in the original domain before the ADMM optimization. For the ADMM-Log-I-ELM algorithm, the learning approach and ADMM optimization process are unchanged, whereas the learning is done in the log domain. Our numerical results show that the MSE of both algorithms converges and, in terms of relative error, the ADMM-Log-I-ELM algorithm is much better than the ADMM-I-ELM algorithm. Furthermore, we conducted several experiments under various settings to compare the estimated values for both proposed approaches. It is noticed that the ADMM-I-ELM algorithm overestimates the blocking value, especially in the low blocking region. However, the ADMM-Log-I-ELM algorithm estimates the blocking probability in an accurate manner. Therefore, training conducted in the log domain can resolve the overestimation problem for those low blocking values. Moreover, the estimation times of our approaches are within a second. Compared to the 20 min runtime of simulation, our approaches are much faster. In addition, we compared our approaches with the well-known analytical approximation EFPA. From our numerical results, our proposed ADMM-Log-I-ELM algorithm was in general much more accurate than EFPA. To sum up, the feasibility of ELM with ADMM on estimating blocking probability was demonstrated in this paper. It is expected that our proposed concept can be easily extended to the other kinds of networks as well.

## Acknowledgment

The work was partially supported by a research grant from the Government of the Hong Kong Special Administrative Region (CityU 11259516).

## References

[1] C. Qiao and M. Yoo, "Optical burst switching (OBS)—A new paradigm for an optical Internet," *J. High Speed Netw.*, vol. 8, no. 1, pp. 69–84, Mar. 1999.

[2] J. S. Turner, "Terabit burst switching," *J. High Speed Netw.*, vol. 8, no. 1, pp. 3–16, Mar. 1999.

[3] K. Dolzer, C. Gauger, J. Späth, and B. Stefan, "Evaluation of reservation mechanisms for optical burst switching," *Int. J. Electron. Commun.*, vol. 55, no. 1, pp. 18–26, Jan. 2001.

[4] G. N. Rouskas and L. Xu, "Optical packet switching," in *Emerging Optical Network Technologies*. Springer, 2005, pp. 111–127.

[5] P. Garg and N. Goyal, "Blocking performance enhancement using congestion control in optical burst switching networks," *Int. J. Electron. Comput. Sci. Eng.*, vol. 1, no. 4, pp. 2217–2220, Sept. 2012.

[6] P. Garg and N. Goyal, "Congestion control using modified Erlangs loss formulae," *Int. J. Eng. Res. Technol.*, vol. 1, no. 6, pp. 1–4, Aug. 2012.

[7] H. Rauthan, A. Verma, and G. Kaur, "Congestion control strategy in optical burst switching networks," *Int. J. Comput. Appl.*, vol. 91, no. 17, pp. 33–37, Apr. 2014.

[8] A. Kaheel, H. Alnuweiri, and F. Gebali, "Analytical evaluation of blocking probability in optical burst switching networks," in *IEEE Int. Conf. on Communications*, 2004, vol. 3, pp. 1548–1553.

[9] C.-C. Hsu, M. Devetsikiotis, and S. D. Roberts, "Fast simulation of optical burst switching networks using simulated annealing," in *14th IEEE Int. Symp. on Modeling, Analysis, and Simulation*, 2006, pp. 283–292.

[10] J. J. P. C. Rodrigues, N. M. Garcia, M. M. Freire, and P. Lorenz, "Object-oriented modeling and simulation of optical burst switching networks," in *IEEE GLOBECOM Workshops*, 2004, pp. 288–292.

[11] Z. Rosberg, H. L. Vu, M. Zukerman, and J. White, "Performance analyses of optical burst-switching networks," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 7, pp. 1187–1197, Sept. 2003.

[12] S. Li, M. Wang, E. W. Wong, H. Overby, and M. Zukerman, "Evaluation of burst/packet loss ratio in a bufferless OBS/OPS network with 1+X path protection," *IEEE Photon. Technol. Lett.*, vol. 28, no. 15, pp. 1688–1691, Aug. 2016.

[13] 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," *J. Opt. Commun. Netw.*, vol. 7, no. 5, pp. 380–391, May 2015.

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

[15] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, Mar. 1991.

[16] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Trans. Inf. Theory*, vol. 39, no. 3, pp. 930–945, May 1993.

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

[18] J. Tang, C. Deng, and G. B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 809–821, Apr. 2016.

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

[20] 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.

[21] 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.

[22] Y. Chen, C. Qiao, and X. Yu, "Optical burst switching: A new area in optical networking research," *IEEE Network*, vol. 18, no. 3, pp. 16–23, May 2004.

[23] T. Venkatesh, C. S. R. Murthy, and C. Murthy, *An Analytical Approach to Optical Burst Switched Networks*. Springer, 2010.

[24] A. K. Garg and R. Kaler, "Burst dropping policies in optical burst switched network," *Optik*, vol. 121, no. 15, pp. 1355–1362, Sept. 2010.

[25] I. Szcześniak, "Overview of optical packet switching," *Theoret. Appl. Inform.*, vol. 21, no. 3–4, pp. 167–180, Nov. 2009.

[26] H. Øverby and N. Stol, "Providing absolute QoS in asynchronous bufferless optical packet/burst switched networks with the adaptive preemptive drop policy," *Comput. Commun.*, vol. 28, no. 9, pp. 1038–1049, June 2005.

[27] H. C. Cho, M. S. Fadali, and H. Lee, "Neural network control for TCP network congestion," in *IEEE Proc. American Control Conf.*, 2005, pp. 3480–3485.

[28] A. A. Al Islam and V. Raghunathan, "ITCP: An intelligent TCP with neural network based end-to-end congestion control for ad-hoc multi-hop wireless mesh networks," *Wireless Networks*, vol. 21, no. 2, pp. 581–610, Feb. 2015.

[29] H. T. Huynh and Y. Won, "Extreme learning machine with fuzzy activation function," in *IEEE 5th Int. Joint Conf. on INC, IMS, and IDC (NCM)*, 2009, pp. 303–307.

[30] C. S. Leung, W. Y. Wan, and R. Feng, "A regularizer approach for RBF networks under the concurrent weight failure situation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 6, pp. 1360–1372, June 2017.

[31] X. Liu, S. Lin, J. Fang, and Z. Xu, "Is extreme learning machine feasible? A theoretical assessment (Part I)," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 7–20, Jan. 2015.

[32] S. Lin, X. Liu, J. Fang, and Z. Xu, "Is extreme learning machine feasible? A theoretical assessment (Part II)," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 21–34, Jan. 2015.

[33] D. Medhi, *Network Routing: Algorithms, Protocols, and Architectures*. Morgan Kaufmann, 2010, chap. 11, pp. 237–345.

[34] R. Lao and R. Killey, "Design of wavelength-routed optical network topologies to minimise lightpath blocking probabilities," in *Proc. London Communications Symp.*, 2003.

[35] S. Li, M. Wang, H. Overby, E. W. Wong, and M. Zukerman, "Performanceevaluation of a bufferless OBS/OPS network with 1 + 1 pathprotection," *IEEE Photon. Technol. Lett.*, vol. 27, no. 20, pp. 2115–2118, Oct. 2015.

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