PID Controller Based on a Self-Adaptive Neural Network to Ensure QoS Bandwidth Requirements in Passive Optical Networks

N. Merayo, D. Juárez, Juan C. Aguado, I. de Miguel, R. J. Durán, P. Fernández, R. M. Lorenzo, and E. J. Abril

***Abstract*—In this paper, a proportional-integral-derivative (PID) controller integrated with a neural network (NN) is proposed to ensure quality of service (QoS) bandwidth requirements in passive optical networks (PONs). To the best of our knowledge, this is the first time an approach that implements a NN to tune a PID to deal with QoS in PONs is used. In contrast to other tuning techniques such as Ziegler–Nichols or genetic algorithms (GA), our proposal allows a real-time adjustment of the tuning parameters according to the network conditions. Thus, the new algorithm provides an online control of the tuning process unlike the ZN and GA techniques, whose tuning parameters are calculated offline. The algorithm, called neural network service level PID (NN-SPID), guarantees minimum bandwidth levels to users depending on their service level agreement, and it is compared with a tuning technique based on genetic algorithms (GA-SPID). The simulation study demonstrates that NN-SPID continuously adapts the tuning parameters, achieving lower fluctuations than GA-SPID in the allocation process. As a consequence, it provides a more stable response than GA-SPID since it needs to launch the GA to obtain new tuning values. Furthermore, NN-SPID guarantees the minimum bandwidth levels faster than GA-SPID. Finally, NN-SPID is more robust than GA-SPID under real-time changes of the guaranteed bandwidth levels, as GA-SPID shows high fluctuations in the allocated bandwidth, especially just after any change is made.**

***Index Terms*—Dynamic bandwidth allocation (DBA); Neural network (NN); Passive optical network (PON); Proportional-integral-derivative (PID); Quality of service (QoS); Service level agreement (SLA).**

I. Introduction

Passive optical networks (PONs) and large-range PONs (LR-PONs) have become the most deployed access network technologies over the world in the last few years. In fact, the report presented in the FTTH Council [1] states that about 22 million homes in Europe will be connected by the end of 2018, amounting to 10.6% of all homes. Moreover, 100 million people in the Asia–Pacific region now subscribe to fiber to the home (FTTH) services by means of PON infrastructures.

In a typical configuration a PON follows a tree topology between the optical line terminal (OLT), located in the central office, and an optical network unit (ONU) located at the user's house (Fig. 1). These access networks use optical fiber as the transmission medium and they are provided with two channels, that is, two independent wavelengths. In the upstream channel (from ONUs to the OLT), PONs have a multipoint-to-point connectivity so every ONU shares the same wavelength. As a consequence, a medium access control protocol is required to manage data collisions of different users (ONUs).

Dynamic bandwidth allocation (DBA) algorithms, based on the time division multiple access protocol, are the most extended option, as they dynamically distribute the available bandwidth depending on the real-time bandwidth demand or the quality of service (QoS) requirements contracted by end users [2–9]. On the other hand, the connectivity in the downstream channel (from the OLT to the ONUs) is point-to-point, so the optical splitter (Fig. 1) broadcasts the input traffic to every ONU.

Regarding QoS, access networks have to differ among priority profiles by means of service level agreements (SLAs) that are contracted by users with service providers. Each profile consists of different services and applications, and the PON should ensure some requirements for each service—typically associated with minimum guaranteed bandwidth levels, maximum delays, or maximum permitted jitters. These QoS restrictions have to be efficiently fulfilled by DBA algorithms, and it is highly desirable that the performance of these algorithms becomes independent from real-time network or traffic conditions.

In this way, the majority of DBA algorithms are focused on complying with QoS requirements [10,11] stipulated by service providers. The most common type of SLA contracts strive to ensure minimum bandwidth levels depending on the profile priority and its related applications or services,

Manuscript received December 16, 2016; revised March 23, 2017; accepted March 23, 2017; published April 26, 2017 (Doc. ID 282806).

The authors are with the Optical Communications Group of the Department of Signal Theory, Communications and Telematic Engineering, E.T.S.I. Telecomunicación, Universidad de Valladolid, Campus Miguel Delibes, Paseo de Belén 15, 47011 Valladolid, Spain (e-mail: noemer@tel.uva.es).

https://doi.org/10.1364/JOCN.9.000433

Authorized licensed use limited to: SHIV NADAR UNIVERSITY. Downloaded on January 16,2023 at 11:41:12 UTC from IEEE Xplore. Restrictions apply.

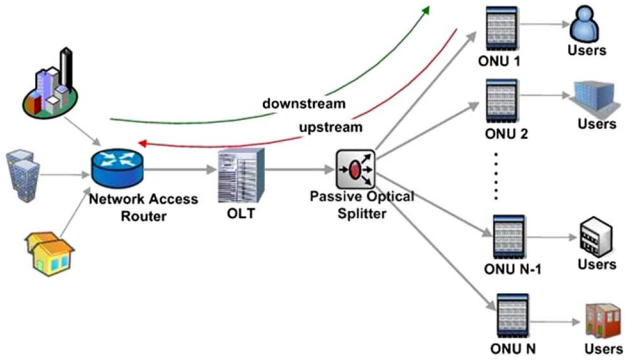

Fig. 1. Typical topology of a PON and LR-PON access infrastructure.

so service providers should be provided with robust DBA algorithms that comply with these guaranteed bandwidths. The authors of Ref. [12] proposed an algorithm that allows bandwidth and delay bounds for guaranteed service traffic, but they do not fulfill the bandwidth for the entire profile. In Ref. [13], the authors proposed a DBA algorithm to provide bandwidth guarantees for only high-priority subscribers based on the contracted SLA while providing best-effort service to the remaining subscribers. The algorithm developed in Ref. [14] guarantees fairness among users by allocating excess bandwidth proportional to their subscription rates. On the other hand, many research algorithms apply fixed weighted factors to each SLA to comply with its associated minimum bandwidth requirements [15,16]. However, these algorithms cannot properly react when real-time changes are made by service providers in the SLA conditions.

Therefore, it is desirable that the DBA algorithms be independent from the initial weights or bandwidth conditions and they be able to automatically adapt to the new network conditions. To overcome this situation, proportional-integral-derivative (PID) systems provide a robust way to automatically control QoS requirements to end users in PONs. PID controllers are able to offer a reliable, stable, and effective response when controlling different systems, and they are used in many environments due to their good performance [17–20]. However, their implementation in optical networks is quite limited. For example, one PID controller was integrated in dynamic optical burst-switched networks [21] to model a wavelength-locking loop in order to stabilize the output of tunable lasers. In Refs. [22,23], the authors designed controllers to manage certain tasks in optical grid networks and for the establishment of light-paths in wavelength division multiplexing networks. The authors of Ref. [24] proposed a GA-based PID active queue management control design for a class of Transmission Control Protocol (TCP) communication networks. Quite recently, we proposed the design of PID and P controllers to efficiently deal with bandwidth and delay requirements in PONs [25,26]. In particular, the algorithm in Ref. [25] proposes a PID that ensures that different profiles are ensured minimum guaranteed bandwidth levels according to their contracted SLA. In Ref. [26], we implemented a P controller that supports QoS requirements regarding maximum delays of highly sensitive traffic. Both strategies demonstrated better performance and robustness than previously existing DBA algorithms.

However, PID systems need to be tuned according to the system under control. A great number of tuning techniques, such as the well-known Ziegler–Nichols (ZN) frequency response method, are manual and based on experiments, so they are laborious, quite slow, and imprecise. As a consequence, many novel tuning approaches implement expert systems techniques to automatically tune PID controllers with high accuracy and robustness [27] because they provide intelligent advice or make intelligent decisions using rule-based programs or expert knowledge in one specific field. Some of these techniques are fuzzy logic, genetic algorithms, artificial intelligence, or neural networks.

On the one hand, fuzzy logic sometimes bases the reasoning and decisions on incomplete information similar to that of human beings [28] and if the system is complex, the selection of the fuzzy rules and functions may be a laborious issue [29,30]. In contrast, genetic algorithms have demonstrated better performance, stability, and robustness than other techniques, providing a range of good solutions for the tuning parameters as genetic algorithms select and reproduce the best individuals of each population [20,31–35]. Consequently, in a recent research study [36], we designed and implemented a tuning technique based on a genetic algorithm integrated in a P controller to deal with QoS delay constraints of high-priority services in PONs. Although the simulation study showed a very good performance to control network parameters, genetic algorithms are an offline technique—which means that if traffic or network conditions change with time, it should be necessary to launch the genetic algorithm to look for the best solutions (tuning parameters) under the new conditions.

Consequently, the integration of online tuning techniques should be quite appealing in solving this problem. Therefore, in this proposal we have selected a neural network to design an online and adaptive tuning process over time [37–39]. However, traditional neural networks (NNs) need to be provided with good examples to be trained, and consequently the learning process can become quite long because their performance highly depends on the amount of data they are trained with [28]. In contrast, our proposal focuses on developing a self-learning and adaptive neural network that permits an online and real-time adjustment of the tuning parameters of the PID according to the recent working status of the network. In fact, there are many algorithms in the literature that implement self-adaptive neural networks to design intelligent PID controllers [40–43]. In particular, we propose to develop an expert and intelligent PID controller based on a NN in order to automatically provide minimum bandwidth levels to different profiles. To the best of our knowledge, this is the first bandwidth-allocation algorithm that implements a NN technique to optimize the tuning process in PID controllers to efficiently provide QoS bandwidth guarantees in PONs.

The rest of this paper is organized as follows. Section II describes the PID controller implemented to provide bandwidth guarantees to different kinds of user profiles. Moreover, the design of the neural network to tune the controller is

Authorized licensed use limited to: SHIV NADAR UNIVERSITY. Downloaded on January 16,2023 at 11:41:12 UTC from IEEE Xplore. Restrictions apply.

explained. Section III shows the simulation results and discusses results. Section IV addresses the conclusions of the paper and some future research proposals.

II. Design of a PID Controller With an Auto-Adaptive Neural Network to Guarantee Minimum Bandwidth Levels to Different-Priority Profiles

A. Principles of Neural Networks

Neural networks are simplified mathematical models that try to emulate the human nervous system, so they try to extract brain capacities to solve complex problems. A neural network is a big mesh that propagates signals from one neuron to others, and it constantly modifies the strength of the response in order to activate or inhibit the connection between neurons. Artificial neural networks work in the same way [44]: each artificial neuron has an internal state, called activation level, so each neuron receives signals that permit it to change its state using the activation function and the received signals. The input of one neuron is calculated as the sum of every input weighted by a factor, as can be observed in Fig. 2.

Therefore, this synaptic weight (positive, negative, zero) defines the strength of the connection between two neurons. For example, a positive weight works as a positive stimulation and a negative weight acts as an inhibitor. As a consequence, by means of the adjustment of the synaptic weights, the neural network adapts its response to different environments. Finally, the input of the neuron is processed by the activation function that provides the output of the neuron, and this output is propagated to the next connected neurons [44]. On the other hand, the organization of the neurons inside the network is the topology, and it depends on the number of layers, number of neurons at each layer, and the degree of connection between neurons. The most common neural networks are multilayer, so the input layer receives information from the outside and the last layer is the final output. Furthermore, there are several intermediate layers called hidden layers (Fig. 2).

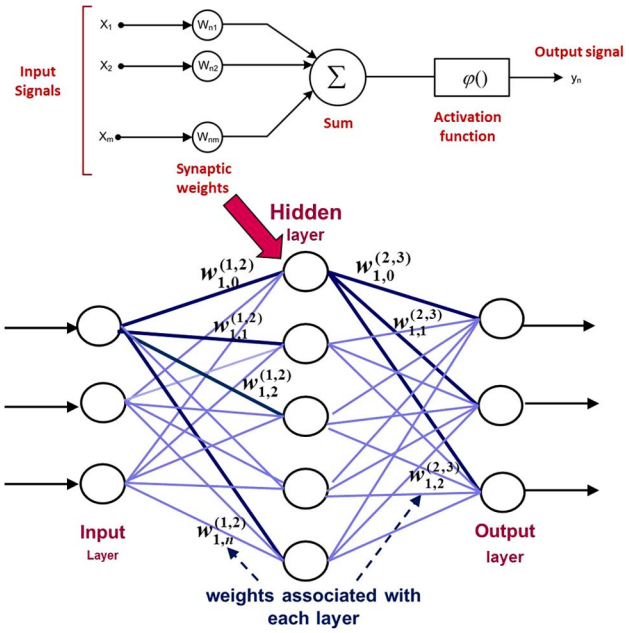

Fig. 2. Structure of one artificial neuron and a neural network.

B. Design of the PID Controller to Ensure QoS Bandwidth Requirements in the OLT

In previous works, we have developed several algorithms based on PIDs to provide QoS requirements in PON networks. In fact, the DBA algorithm presented in [25], called service level agreement PID (SPID), efficiently assigns minimum bandwidth levels using a PID controller. The main purpose of PID controllers and its variants (P and PI controllers) is to keep the value of a variable close to a reference value [18,45]. Thus, the controller calculates the error, defined as the difference between the current value of the variable under control and its reference value. According to this committed error ($e[n]$), the PID updates one control signal ($u[n]$), where

$$u[n] = K_p \cdot e[n] + K_p \cdot \frac{T_{\text{sample}}}{T_i} \sum_{m=0}^{n} e[n] + K_p \cdot \frac{T_d}{T_{\text{sample}}} (e[n] - e[n-1]). \quad (1)$$

This control signal is composed of the proportional term (as regards the current error), the integral term (accumulation of past errors), and the derivative term (prediction of future errors). Moreover, the variable K_p is the proportional gain, T_i is the integral gain, and T_d is the derivative gain.

To assign bandwidth to each ONU_i at every cycle ($B_{\text{alloc}}^{\text{onu}_i}$), the algorithm implements a polling policy with a limited scheme—the most common and efficient way to allocate bandwidth in PON networks [46]. This limited and simple scheme is defined as $B_{\text{alloc}}^{\text{onu}_i} = \text{Minimum}\{B_{\text{required}}^{\text{onu}_i}, B_{\text{max}}^{\text{onu}_i}\}$, where $B_{\text{required}}^{\text{onu}_i}$ is the bandwidth demanded by ONU_i in one cycle and $B_{\text{max}}^{\text{onu}_i}$ is the maximum bandwidth allowed to each ONU at one cycle (to avoid the channel monopolization of some ONUs). In order to integrate the controller in the bandwidth-allocation process ($B_{\text{alloc}}^{\text{onu}_i}$), the term $B_{\text{max}}^{\text{onu}_i}$ is constantly updated by means of the control signal ($u[n]$) so that $B_{\text{max}}^{\text{onu}_i}$ can dynamically evolve to efficiently ensure the minimum bandwidth levels to every profile.

The block diagram of the algorithm is shown in Fig. 3. As the PID algorithm controls the guaranteed bandwidth associated with every ONU, the reference value is the minimum guaranteed bandwidth that has to be ensured by the service provider depending on the contracted SLA ($B_{\text{guaranteed}}^{\text{sla}\in\text{onu}_i}$). The term under control is the mean allocated bandwidth associated with each ONU_i ($\overline{B_{\text{alloc}}^{\text{onu}_i}[n]}$), so to calculate the instantaneous error ($e[n]$) committed at each ONU, it is necessary to subtract the mean allocated bandwidth from its guarantee bandwidth level ($e[n] = B_{\text{guarantee}}^{\text{sla}\in\text{onu}_i} - \overline{B_{\text{alloc}}^{\text{onu}_i}}$).

Authorized licensed use limited to: SHIV NADAR UNIVERSITY. Downloaded on January 16,2023 at 11:41:12 UTC from IEEE Xplore. Restrictions apply.

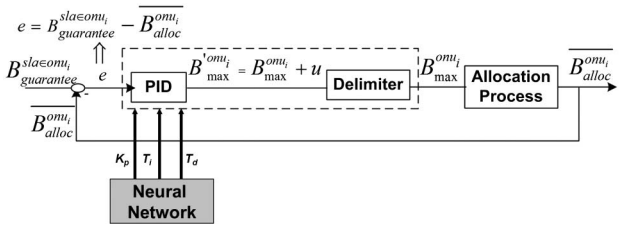

Fig. 3. Block diagram of the designed PID to control the QoS bandwidth requirements in PONs.

To update the maximum permitted bandwidth ($B_{\max}^{\mathrm{onu}_i}$), the control signal $u[n]$ is added to the previous maximum permitted bandwidth ($B_{\max}^{\prime\mathrm{onu}_i} = B_{\max}^{\mathrm{onu}_i} + u[n]$), as can be observed in Fig. 3. In this way, if, for example, the mean allocated bandwidth of ONU_i is lower than their guaranteed bandwidth level, the error becomes positive and the control signal as well, so the algorithm increments its maximum permitted bandwidth to allow ONU_i to comply with the QoS bandwidth restrictions. Finally, a delimiter is included in the system to adapt the maximums proportionally to those calculated by the controller so as to fit in the maximum cycle time of the EPON standard (2 ms) (Fig. 3).

C. Design of the Neural Network to Tune the PID Controller

To calculate the tuning parameters of the PID controller, the algorithm SPID uses the frequency response method proposed by Ziegler–Nichols [18,45]. However, ZN is a manual technique that may become time-consuming and laborious if the selected values are far from the suitable ones, as demonstrated in Ref. [36]. In contrast, in Ref. [36] we enhanced the performance of SPID by incorporating a genetic algorithm to automate the tuning process—the so-called genetic algorithm SPID (GA-SPID). Although the GA efficiently automates the tuning process and these tuning parameters offer good performance in the bandwidth-allocation process, this technique does not permit a real-time adaptation of the tuning parameters in case the network or traffic conditions suddenly change. Therefore, genetic algorithms are considered an offline tuning technique.

However, it is quite desirable to design one method that updates the tuning parameters according to whatever changes inside the network, that is, an online tuning technique. In this way, we propose the integration of an auto-adaptive neural network inside the PID controller that auto-learns from the last committed errors depending on the real-time network conditions, called a neural network SPID (NN-SPID) algorithm. To design a neural network that tunes the controller, we apply one alternative formula for the control signal of the PID in discrete time. This expression is represented in Eq. (2), where K_p is the proportional constant, K_i is the integral constant, and K_d is the derivative constant. In this way, the tuning parameters using neural networks will be K_p, K_i, and K_d. However, the relationship between these parameters and the T_i and T_d terms of Eq. (1) is given by Eq. (3), where T is the sample time of the PID (time between two consecutive executions of the PID):

$$\Delta u(k) = \Delta u[k-1] + K_p(e[k] - e[k-1]) + K_i e[k] + K_d(e[k] - 2e[k-1] + e[k-2]), \tag{2}$$

$$K_i = K_p \frac{T}{T_i}, \qquad K_d = K_p \frac{T_d}{T}. \tag{3}$$

Therefore, the general scheme of the controlled process using a NN is shown in Fig. 4. As can be observed in the picture, the tuning parameters K_p, K_i, and K_d are provided by the output of the neural network. Furthermore, our proposed method needs to know the real-time error of the PID controller [Eq. (2)] to efficiently update the tuning parameters. Consequently, this error is provided by the PID controller and so both of them are constantly interchanging information inside the OLT, as can be noticed in Fig. 3. In fact, this error corresponds to the absolute value of the difference between the guaranteed bandwidth of each ONU ($B_{\mathrm{guaranteed}}^{\mathrm{sla}\in\mathrm{onu}_i}$) and its mean allocated bandwidth ($\overline{B_{\mathrm{alloc}}^{\mathrm{onu}_i}}[n]$), that is, $e[k] = |B_{\mathrm{guarantee}}^{\mathrm{sla}\in\mathrm{onu}_i} - \overline{B_{\mathrm{alloc}}^{\mathrm{onu}_i}}[n]|$. Finally, it is worth saying that the PID controller and the NN are integrated in the OLT since the bandwidth-allocation process of the overall EPON is carried out inside this intelligent component.

On the other hand, to design the architecture of the neural network, it is necessary to select some parameters such as the number of layers, neurons, activation functions, and training methods. Regarding the number of layers, the literature reveals that many algorithms implement one hidden layer (apart from the input and the output layer) because it provides a simple structure and good performance [37–39]. The number of neurons at the output layer accords with the tuning parameters of the PID, so it is set to three (K_p, K_i, and K_d), and the number of neurons in the input layer is three because the tuning parameters are calculated based on the three last errors $e[k]$, $e[k-1]$, and $e[k-2]$ [following Eq. (2)]. Finally, the number of neurons in the hidden layer depends on the accuracy we want to achieve, but the literature points out that a number between one and four times the number of inputs is enough; therefore, we select five neurons (simulations have been done to demonstrate this selection) [37–39].

Moreover, we have added a bias signal at the intermediate and output layers to look for a more stable response of

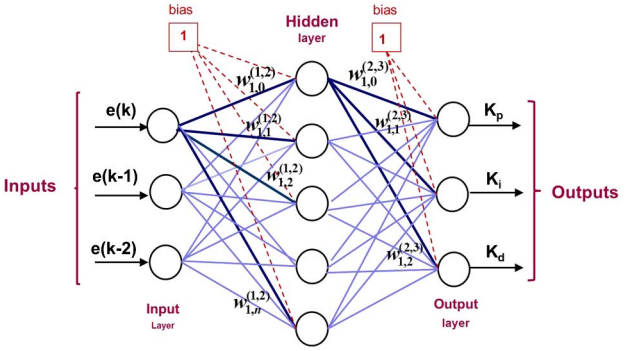

Fig. 4. Flow diagram of the designed neural network to control QoS bandwidth requirements in NN-SPID.

Authorized licensed use limited to: SHIV NADAR UNIVERSITY. Downloaded on January 16,2023 at 11:41:12 UTC from IEEE Xplore. Restrictions apply.

the system. Then, the feedforward artificial neural network is a multilayer perceptron that maps sets of input data onto a set of appropriate outputs, that is, good tuning parameters according to the errors (Fig. 4). Furthermore, the selection of the activation function is based on the range of values of the desirable results. In this way, the sigmoid function is chosen for the input layer since it is one of the most classical and extended. For the output layer, although the sigmoid function is the most used, we select the lineal function since we need a large number of values for the tuning parameters (K_p, K_i, and K_d) and the values of the sigmoid function are limited by a lower and maximum threshold.

Traditional neural networks use learning techniques in which the network is provided with a set of good examples to efficiently learn and work, and they keep weights between neurons with fixed values. In contrast, our proposal constantly auto-learns from the last errors (applying an incremental learning technique), so it constantly changes the weights between neurons. The mathematical study used to increment the matrix of weights is based on several published studies [40–43].

In this way, the backpropagation NN is one of the most implemented neural networks in many systems. It uses the backpropagation learning algorithm applied in two stages: data feedforward and error backpropagation. In the data feedforward phase, the error of the PID controller ($e[k], e[k-1], e[k-2]$) is inserted in the input layer and propagated to the next layers (hidden and output layers), and the algorithm obtains the new tuning parameters for the next cycles. On the other hand, in the error backpropagation stage, with the last output values (tuning parameters), the committed error is calculated (typically using the quadratic performance index) and the values are propagated backward (from output to input). Therefore, the weights between neurons are adjusted using some method (typically the gradient descent algorithm) based on these errors.

These two phases are continuously repeated along the time to ensure a good performance of the bandwidth-allocation process. In our proposal, at each number of iterations of the PID, called n, the tuning parameters are updated (feedforward algorithm). Further, at each number of iterations of the PID (called m), the matrix of weights is updated (backpropagation algorithm). The value of m has to be higher than n because the tuning parameters have to be updated, taking into account the new matrix of weights. If not, periods may exist with changes in the matrix of weights that are not reflected in the calculation of the new tuning parameters. In order to better follow the mathematical approach used in our proposal, Table I summarizes the notations and definitions regarding the NN and the feedforward-propagating and backpropagation algorithms.

In particular, the feedforward-propagating algorithm calculates the input and output values of each layer according to the current weights and the last inputs of the NN (the last three errors and the bias signal), achieving the new tuning parameters at the output of the NN. Then, the input layer follows

$$O_j^{(1)}[k] = x[k] \qquad (j = 0, 1, \ldots), \tag{4}$$

TABLE I
TABLE OF NOTATIONS REGARDING THE NN AND THE FEEDFORWARD-PROPAGATING AND BACKPROPAGATION ALGORITHMS

Notation	Definition
k_p, k_i, k_d	Tuning parameters of the PID
$e[k] = \lvert B^{\text{sla}\in\text{onu}_i}_{\text{guarantee}} - \overline{B^{\text{onu}_i}_{\text{alloc}}[n]}\rvert$	Committed error; difference between the guaranteed bandwidth of each ONU and its mean allocated bandwidth
n	Number of iterations when the tuning parameters are updated
m	Number of iterations when weights of the NN are updated
$I_i^{(L)}[k]$	Input of neuron i at layer L
$O_i^{(L)}[k]$	Output of neuron i at layer L
$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	Sigmoid function; activation function for the hidden layer
$g(x) = x$	Linear function; activation function for the output layer
$\Delta w_{li}^{(a,b)}[k]$	Increments in the last weight modification between neuron i at layer a and neuron j at layer b
$J_1 = (\frac{1}{2})e^2[k]$	Quadratic performance index of error, used to modify the network weights
α	Inertia term in the NN
η	Learning rate coefficient in the NN

in which the inputs $x[k]$ correspond to $e[k], e[k-1], e[k-2]$, and the bias input and $O_j^{(1)}[k]$ is the output of the layer. The superscripts in the different Eqs. (1)–(3) are associated with the layer, that is, input, hidden, and output layer. Moreover, the superscript of the weights refers to weights between two layers; for example; $w_{ij}^{(1,2)}$ corresponds to the weights between the input and the hidden layer.

For the hidden layer (subscript 2), the inputs ($I_i^{(2)}[k]$) and outputs ($O_i^{(2)}[k]$) follow

$$I_i^{(2)}[k] = \sum_j w_{ij}^{(1,2)} O_j^{(1)}[k], \tag{5}$$

$$O_i^{(2)}[k] = f(I_i^{(2)}[k]) \qquad (i = 0, 1, \ldots). \tag{6}$$

Here, $w_{ij}^{(1,2)}$ are the weights between the input and hidden layers. As was said before, the sigmoid function (with positive and negative symmetry) is chosen as the activation function for this layer ($f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$).

Finally, the inputs and outputs of the output layer follow Eqs. (7) and (8). The output of this layer corresponds with the three tuning parameters (K_p, K_i, and K_d) and the activation function for this layer is the linear function ($g(x) = x$). However, the lineal function is truncated for avoiding negative values since tuning parameters have to be positive. The terms $w_{ij}^{(2,3)}$ are the weights between the hidden layer and the output layer:

Authorized licensed use limited to: SHIV NADAR UNIVERSITY. Downloaded on January 16,2023 at 11:41:12 UTC from IEEE Xplore. Restrictions apply.

$$I_l^{(3)}[k] = \sum_i w_{li}^{(2,3)} O_i^{(2)}[k], \tag{7}$$

$$O_i^{(3)}[k] = g(I_l^{(3)}[k])(l = 0, 1, \ldots),$$

so

$$O_1^{(3)}[k] = k_p,$$
$$O_2^{(3)}[k] = k_i,$$
$$O_3^{(3)}[k] = k_d. \tag{8}$$

On the other hand, the backpropagation algorithm modifies the weighted factors in the NN, starting from the output layer to the input layer. To update the weights in the neural network, the quadratic performance index of error [Eq. (9)] is used, based on the parameter under control (guaranteed bandwidth levels). Furthermore, weights are modified using the gradient descent method. Then, the increments are adjusted, according to the negative gradient direction, to the weighting coefficients [Eq. (10)], so that an inertia term (α) and learning rate coefficient (η) directly impact the evolution and convergence speed of the NN. Although both parameters are initially set to 0.1 (following the literature) in the results section, we will analyze their impact on the NN. Furthermore, $\Delta W_{li}^{(3)}[k]$ denotes the last increments achieved in the last weight modification between the hidden and the output layers:

$$J_1 = \left(\frac{1}{2}\right) e^2[k] = \left(\frac{1}{2}\right) (B_{\text{guarantee}}^{\text{sla}\in\text{onu}_i} - B_{\text{alloc}}^{\text{onu}_i})^2, \tag{9}$$

$$\Delta w_{li}^{(2,3)}[k] = -\eta \frac{\partial J_1}{\partial w_{li}^{(2,3)}} + \alpha \Delta w_{li}^{(2,3)}[k-1]. \tag{10}$$

To calculate $\frac{\partial J_1}{\partial w_{li}^{(2,3)}}$, we follow

$$\frac{\partial J_1}{\partial w_{li}^{(2,3)}} = \frac{\partial J_1}{\partial y[k]} \frac{\partial y[k]}{\partial \Delta u[k]} \frac{\partial \Delta u[k]}{\partial O_l^{(3)}[k]} \frac{\partial O_l^{(3)}[k]}{\partial I_l^{(3)}[k]} \frac{\partial I_l^{(3)}[k]}{\partial w_{li}^{(2,3)}}. \tag{11}$$

In this equation, the term $\frac{\partial y[k]}{\partial \Delta u[k]}$ is unknown so we replace it by the $\text{sgn}(\frac{\partial y[k]}{\partial \Delta u[k]})$ approximation (the lack of accuracy can be rectified by η) as it is considered in several studies [40–43].

Taking into account Eqs. (2) and (8), we obtain the expressions for $\frac{\partial \Delta u[k]}{\partial O_l^{(3)}[k]}$:

$$\frac{\partial \Delta u[k]}{\partial O_1^{(3)}[k]} = \frac{\partial \Delta u[k]}{\partial K_p} = e[k] - e[k-1],$$
$$\frac{\partial \Delta u[k]}{\partial O_2^{(3)}[k]} = \frac{\partial \Delta u[k]}{\partial K_i} = e[k],$$
$$\frac{\partial \Delta u[k]}{\partial O_3^{(3)}[k]} = \frac{\partial \Delta u[k]}{\partial K_d} = e[k] - 2e[k-1] + e[k-2]. \tag{12}$$

Finally, with these expressions, we can simplify the formula for the weighted factors $\Delta w_{li}^{(2,3)}[k]$ of the output layer as follows:

$$\Delta w_{li}^{(2,3)}[k] = \eta \delta_l^{(3)} O_i^{(2)}[k] + \alpha \Delta w_{li}^{(2,3)}[k-1],$$
$$\text{where } \delta_l^{(3)} = e[k] \text{sgn}\left(\frac{\partial y[k]}{\partial \Delta u[k]}\right) \frac{\partial \Delta u[k]}{\partial O_l^{(3)}[k]} g'[I_i^{(3)}[k]]. \tag{13}$$

In the same way as the output layer, we modify the weighted factors at the hidden layer, following the expressions

$$\Delta w_{ij}^{(1,2)}[k] = \eta \delta_i^{(2)} O_j^{(1)}[k] + \alpha \Delta w_{ij}^{(1,2)}[k-1],$$
$$\delta_i^{(2)} = f'[I_i^{(2)}[k]] \sum_{l=1}^{3} \delta_l^{(3)} w_{li}^{(2,3)}[k]. \tag{14}$$

III. Simulation Results

A. Simulation Scenario

Simulations have been carried out in an Ethernet passive optical network (EPON) (based on the Ethernet protocol) with 16 ONUs and one user connected to each ONU using OMNeT++ 4.2 [47]. The network scenario has similar characteristics proposed by other DBA algorithms in PONs [2–9]. Regarding the packet and traffic distribution, self-similar traffic based on alternating Pareto-distributed on/off periods with a Hurst parameter of $H = 0.8$ is used to emulate the practical Internet traffic, using the self-similar traffic generator developed in [48]. Then, Ethernet packets between 64 and 1518 bytes and symmetry in the traffic load of every ONU it is assumed, as with the majority of existing DBA algorithms [16,21].

In this research, we compare the performance of GA-SPID and NN-SPID, that is, an offline and online tuning technique, respectively. Moreover, both algorithms (GA-SPID and NN-SPID) dynamically modify the maximum permitted bandwidth of every profile (SLAs) depending on the committed error when guaranteeing the minimum stipulated bandwidth levels to each of them. Therefore, three SLAs (SLA_0, SLA_1, SLA_2) in the EPON network and different QoS minimum bandwidth levels to each profile are assumed in order to check the algorithms' performance. However, the main scenario considers 100 Mbps for the SLA_0 profile, 75 Mbps for the SLA_1 profile, and 50 Mbps for the SLA_2 profile. Finally, results are contained within 95% of confidence level.

B. Selection of Parameters for the Neural Network

The first simulation study regards the selection of some parameters to model the neural network. This selection has been done by running simulations and choosing the values that allow good performance. The first parameter to analyze is the number of neurons in the hidden layer, so Table II collects the standard deviation of the mean allocated bandwidth of each ONU over its guaranteed bandwidth in Mbps (ONUs of the same SLA show the same performance). The standard deviation provides a good measure of the algorithm's performance, as low deviations imply

Authorized licensed use limited to: SHIV NADAR UNIVERSITY. Downloaded on January 16,2023 at 11:41:12 UTC from IEEE Xplore. Restrictions apply.

TABLE II
STANDARD DEVIATION OF $\overline{B_{\text{alloc}}^{\text{onu}_i}}$ OVER $B_{\text{guarantee}}^{\text{sla}\in\text{onu}_i}$ IN MBPS CONSIDERING DIFFERENT NUMBERS OF NEURONS IN THE HIDDEN LAYER

Number of Neurons	Standard Deviation SLA_0 (Mbps)	Standard Deviation SLA_1 (Mbps)	Standard Deviation SLA_2 (Mbps)
2	0.046169	0.0378122	0.02310
4	0.04882	0.030392	0.023456
5	0.0497220	0.0297659	0.0234553
6	0.067423	0.02963	0.02432
8	0.05567	0.028723	0.022764

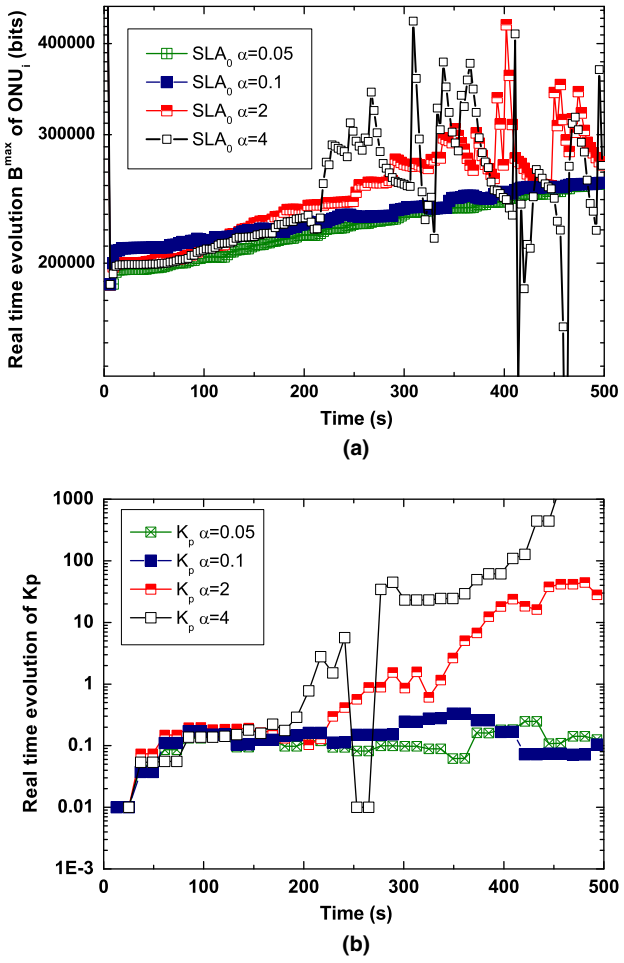

Fig. 5. Parameters of the neural network in NN-SPID with different α values. (a) Real-time evolution of the maximum permitted bandwidth. (b) Real-time evolution of K_p.

high accuracy. Then, the collected data in Table II show that results considering different numbers of neurons are quite similar and very low—less than 0.05 Mbps. As the literature points out that a number between 1 and 4 times the number of inputs (three) is enough, we chose an intermediate value of five neurons that also allows a very low deviation.

On the other hand, another issue to analyze is the evolution speed of the matrix of weights, as can be noticed in

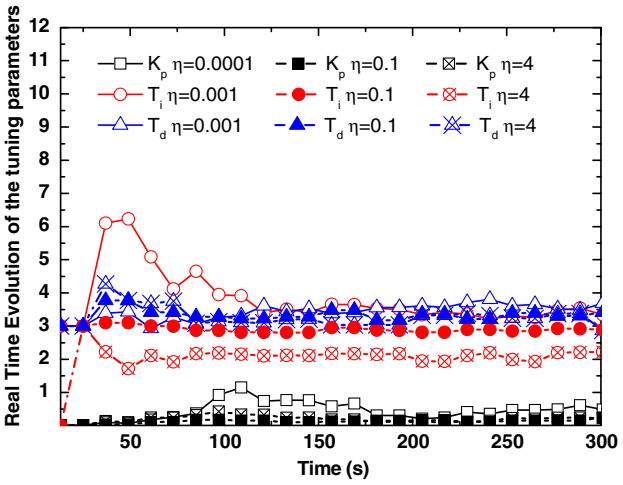

Fig. 6. Real time evolution of the K_p, T_i, and T_d tuning parameters considering different values of η.

Eq. (13). As the goal is to minimize the objective function of the problem, there is a tradeoff between some parameters (α, η), because low values allow a slow convergence to the objective but very high steps can allow the problem to not properly converge to the correct values. The inertia coefficient α in Eq. (13) highly affects the convergence speed in the equation, since it multiplies the previous increment of the weighted factors.

Then, Figs. 5(a) and 5(b) show the impact of this parameter in the evolution of the maximum permitted bandwidth periodically updated by the PID controller and in the real-time evolution of the tuning parameters (we select K_p to reduce the number of graphs). As can be observed in the graphs, high values of α ($\alpha = 2, 4$) allow a bad convergence response of the PID controller that manages the bandwidth-allocation process to efficiently ensure the minimum bandwidth levels [Fig. 5(a)].

This same performance (high fluctuations and incorrect convergence) is translated into the evolution of the K_p values, as can be observed in [Fig. 5(b)]. On the other hand, low values of the inertia coefficient ($\alpha = 0.05, 0.1$) provide a more stable PID response and lower oscillations of the tuning parameter K_p. Therefore, as many research works use a typical value of 0.1, we select the same value [40–43].

On the other hand, Fig. 6 shows the impact of the learning coefficient η in the real-time evolution of the tuning parameters (assuming $\alpha = 0.1$). As can be observed in the graph, the performance of η does not highly impact on the evolution of the tuning parameters as the α parameter does. However, more oscillations can be noticed for very low values ($\eta = 0.0001$) and very high values ($\eta = 4$), so we select an intermediate value of 0.1 that is also adopted in many research works [40–43].

C. Performance of the NN-SPID Algorithm in Comparison With GA-SPID

One of the most important characteristics in both algorithms is the convergence speed to the calculated good

Authorized licensed use limited to: SHIV NADAR UNIVERSITY. Downloaded on January 16,2023 at 11:41:12 UTC from IEEE Xplore. Restrictions apply.

tuning parameters. Regarding the genetic algorithm, the number of iterations of the PID to update the fitness value, population size, and number of generations of the stop criterion have to be fixed in order to simultaneously allow good performance and short tuning time. As was said before, the genetic algorithm is offline, so once it calculates the best tuning parameters, the PID starts to work with these values. The tuning time employed by the genetic algorithm is given by

$$T_{\text{tuning}} = N \cdot m \cdot T_{\text{sample}} \cdot N_{\text{Gen}}, \tag{15}$$

where N is the population size of the genetic algorithm, m is the number of iterations of the PID in which each individual is tested to obtain its fitness, T_{sample} is the sample time used by the PID controller to calculate the committed error, and N_{Gen} is the number of generations of the stop criterion.

One of the parameters to analyze the performance of the tuning time in GA-SPID is T_{sample}. Figure 7(a) shows the real-time evolution of the mean allocated bandwidth when assuming a set of T_{sample} values for the SLA_2 profile (the performance is the same for SLA_0 and SLA_1). As can be noticed in the figure, if T_{sample} is high (30 s), the evolution to the guaranteed bandwidth requirements for this profile, 50 Mbps, becomes slower and unstable. In contrast, a low value for T_{sample} (50 ms) makes the adaptation difficult to control, as the PID lacks enough samples to properly update the error to evolve to the guaranteed bandwidth levels. Therefore, intermediate values (2 or 3 s) show a robust and fast achievement of the stipulated bandwidth requirements and, as we desire a low tuning time, we chose $T_{\text{sample}} = 2$ s.

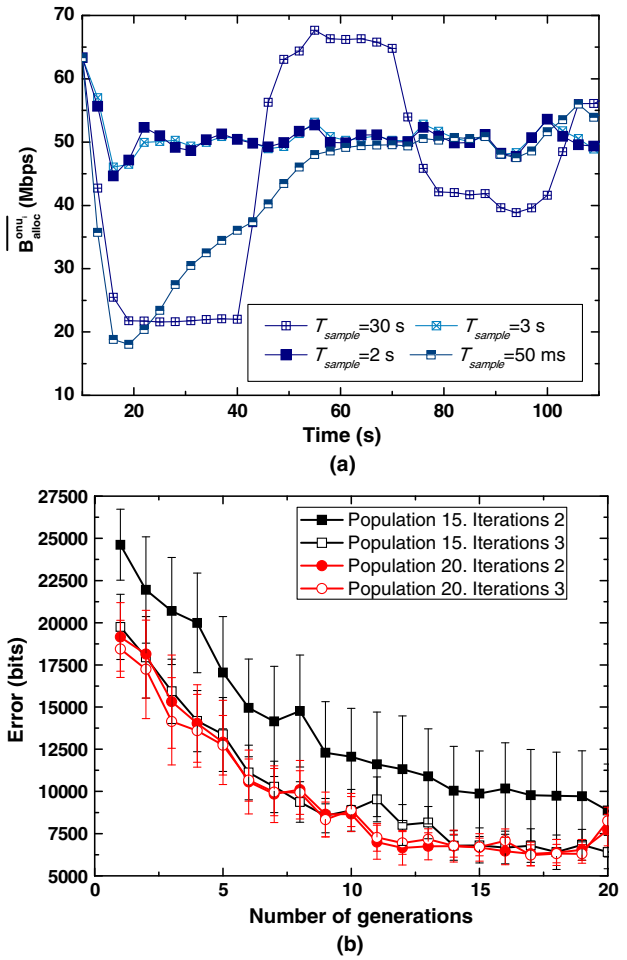

Fig. 7. Parameters of the genetic algorithm in GA-SPID. (a) Evolution of the mean allocated bandwidth when assuming different values of T_{sample}. (b) Evolution of the mean committed error of the best individual of each generation when considering different population sizes and numbers of PID iterations.

Once we select the minimum and efficient value of T_{sample}, we have to determine the remaining parameters that directly impact the tuning time. Then, Fig. 7(b) represents the mean committed error (in bits) of the best individual in each generation when considering different population sizes and numbers of iterations. As was described in depth in [36], GA-SPID follows the main steps of genetic algorithms. Then, each chromosome corresponds with the three tuning parameters (K_p, T_i, T_d) coded in a binary chain (16 bits per parameter). Furthermore, we have to evaluate each individual (chromosome) using an objective function in order to calculate the fitness of each of them. This objective function is based on the committed error of the PID using the individual (K_p, T_i, T_d) during m iterations of the PID, defined as $F = \frac{1}{m} \cdot \frac{1}{N_{\text{onus}}} \sum_m \sum_{i=0}^{N_{\text{onus}}} |e_i[m]|$.

As a consequence, if we observe Fig. 7(b), the committed error is reduced as long as the number of generations increases for every combination of population size and number of iterations. However, the worst performance is achieved for a population of 15 individuals and two iterations. For the other combinations, the results are quite similar. Thus, as we require the lowest tuning time, we select a population of 20 individuals with two iterations. Furthermore, as the committed error becomes quite stable for a number of generations higher than 10, we can select this value for our genetic algorithm. As a consequence, taking into account these previous values and substituting them into Eq. (15), GA-SPID provides a total tuning time (T_{tuning}) equal to 800 s.

In contrast, our new proposal based on neural networks, NN-SPID, permits an online modification of the tuning parameters because it constantly adapts these values according to the network state to ensure the minimum guaranteed bandwidth levels to every profile.

In order to show the tuning time required by NN-SPID to achieve the guaranteed bandwidth levels, Figs. 8(a) and 8(b) represent the real-time evolution of the mean allocated bandwidth for every profile. We check the performance for two different guaranteed bandwidth scenarios (Scenario 1: $\text{SLA}_0 = 100$ Mbps, $\text{SLA}_1 = 75$ Mbps, $\text{SLA}_2 = 50$ Mbps; Scenario 2: $\text{SLA}_0 = 80$ Mbps, $\text{SLA}_1 = 60$ Mbps, $\text{SLA}_2 = 60$ Mbps). For both scenarios, it can be noticed that NN-SPID is faster than SPID since it only needs around 100 s to ensure the minimum bandwidth levels for every profile in Scenario 1 [Fig. 8(a)] and less than 50 s for Scenario 2 [Fig. 8(b)]. In Scenario 1, SLA_0 is given around 90 Mbps since its load is 0.9 (90 Mbps), so NN-SPID gives all demanded bandwidth to this profile. On the contrary,

Authorized licensed use limited to: SHIV NADAR UNIVERSITY. Downloaded on January 16,2023 at 11:41:12 UTC from IEEE Xplore. Restrictions apply.

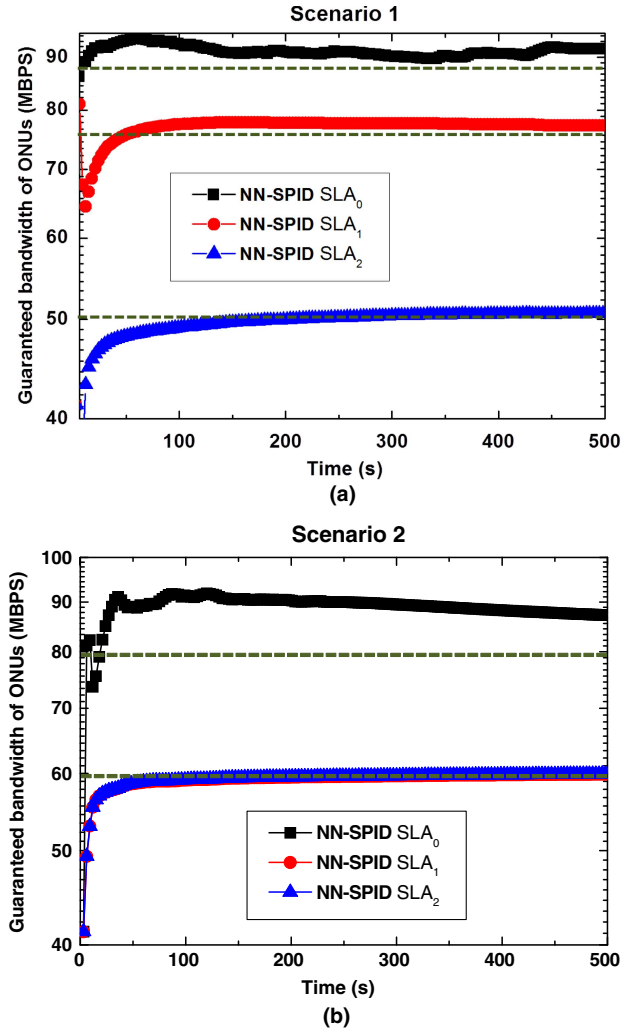

Fig. 8. Real-time evolution of the mean allocated bandwidth of NN-SPID: (a) scenario 1, (b) scenario 2.

GA-SPID requires 800 s to provide good individuals to tune the PID. Consequently, it is demonstrated that NN-SPID provides QoS bandwidth requirements faster than GA-SPID independently from the considered bandwidth scenario.

Furthermore, Figs. 9(a)–9(c) show the real-time evolution of the tuning parameters (K_p, T_i, T_d) for NN-SPID and GA-SPID in order to observe the differences between an adaptive and not adaptive technique. It can be noticed that NN-SPID periodically adapts the tuning parameters, so it modifies the control signal of the PID to face real-time changes in the network conditions. In contrast, the genetic algorithm does not modify the tuning parameters and cannot immediately react when changes in the network conditions may appear.

On the other hand, to compare the robustness of both algorithms, Fig. 10 represents the real-time evolution of the maximum permitted bandwidth (periodically updated by the PID) when comparing NN-SPID and GA-SPID assuming different good individuals for GA-SPID (Scenario 1: $k_p = 1$, $T_i = 2$, $T_d = 5$; Scenario 2: $k_p = 1$, $T_i = 2$, $T_d = 2$). In order to simplify the number of graphs, we only represent SLA_1 and SLA_2, although the performance is similar for SLA_0.

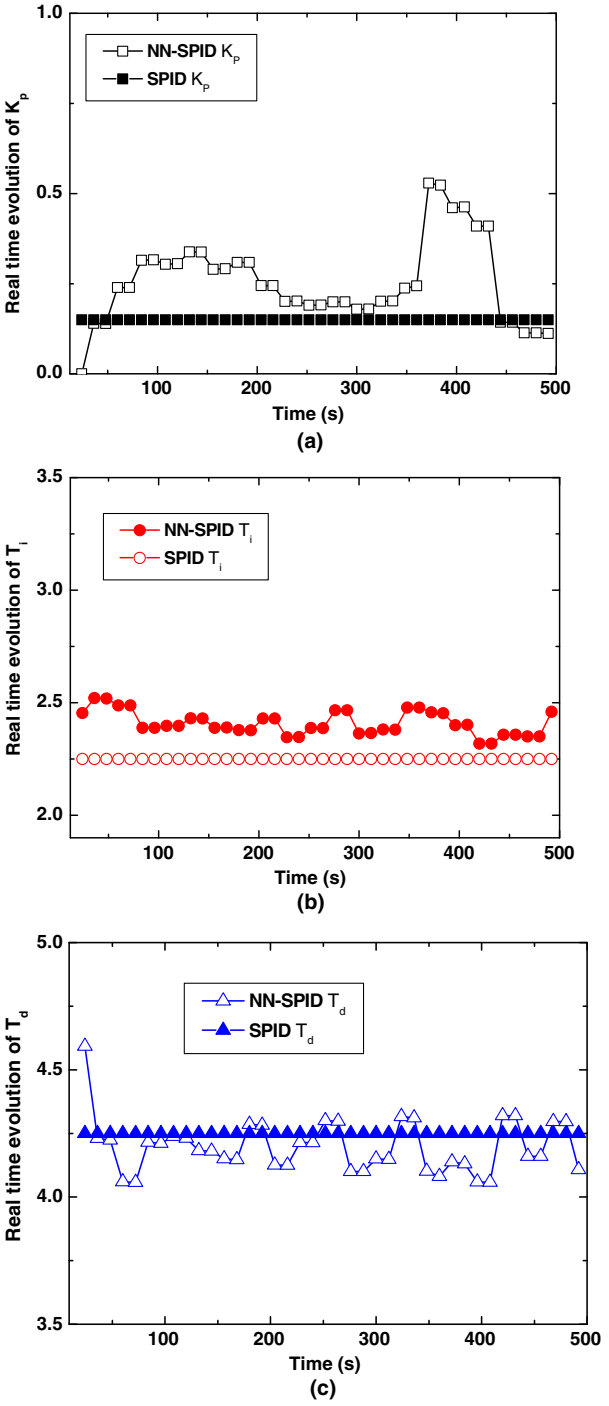

Fig. 9. Real-time evolution of the tuning parameters for NN-SPID and GA-SPID: (a) K_p, (b) T_i, (c) T_d.

As can be noticed in the graph, NN-SPID shows lower oscillations than GA-SPID for both profiles, minimizing every abrupt fluctuation that appears in GA-SPID. This performance demonstrates that NN-SPID provides a more stable and robust response in the bandwidth-allocation

Authorized licensed use limited to: SHIV NADAR UNIVERSITY. Downloaded on January 16,2023 at 11:41:12 UTC from IEEE Xplore. Restrictions apply.

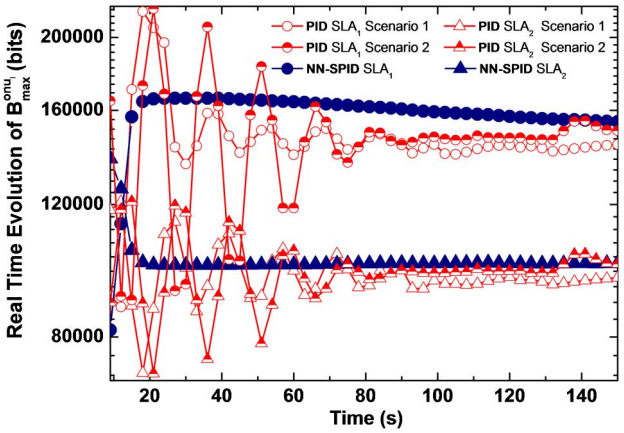

Fig. 10. Real-time evolution of the maximum permitted bandwidth for SLA_1 and SLA_2, comparing NN-SPID and GA-SPID.

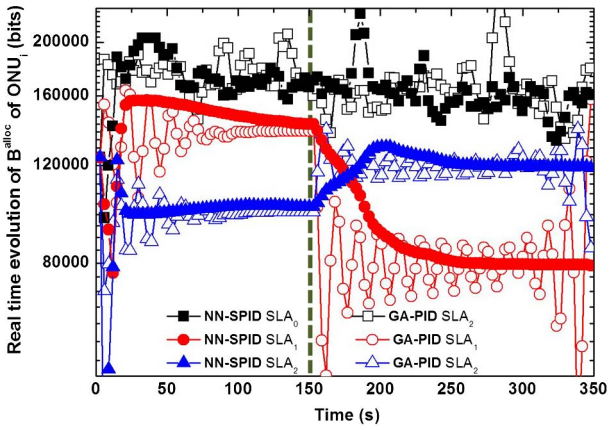

Fig. 11. Variation of the mean allocated bandwidth of every profile when the guaranteed bandwidth levels vary over time.

TABLE III
DIFFERENT GUARANTEED BANDWIDTH LEVELS FOR EVERY PROFILE OVER TIME

Time (s)	Guaranteed Bandwidth SLA_0 (Mbps)	Guaranteed Bandwidth SLA_1 (Mbps)	Guaranteed Bandwidth SLA_2 (Mbps)
0–150	100	70	50
Higher than 150	70	40	60

process than GA-SPID since the neural network applies a real-time reconfiguration of the tuning parameters according to the network state. In contrast, the genetic algorithm has to be periodically launched to update the tuning parameters according to the last network conditions, and this process can take a long time (around 800 s).

Finally, the speed and stability of NN-SPID and GA-SPID become very important when service providers require real-time changes in the guaranteed bandwidth levels. To demonstrate the performance of both algorithms in this situation, simulations have been carried out changing the guaranteed bandwidth levels at 150 s (Table III).

Hence, Fig. 11 shows the evolution of the maximum permitted bandwidth with time, for the considered guaranteed bandwidth levels. As can be noticed, both algorithms dynamically modify the maximum permitted bandwidth so that values can evolve to the new guaranteed bandwidth levels. However, the stability of both algorithms is quite different since GA-SPID shows high fluctuations in the allocation process when the guaranteed bandwidth levels change at 150 s, especially for the lowest-priority profile SLA_2. On the contrary, NN-SPID exhibits a more stable response in the evolution of the maximum permitted bandwidth than GA-SPID, since it makes a real-time adaptation of the tuning parameters according to the real-time bandwidth requirements.

Finally, the effectiveness of the proposed technique was demonstrated in EPONs of high coverage. In fact, we ran simulations in a long-reach EPON of 100 km considering the bandwidth conditions of Scenario 1 ($SLA_0 = 100$ Mbps, $SLA_1 = 75$ Mbps, $SLA_2 = 50$ Mbps). As can be noticed, Fig. 12 compares the performance of NN-SPID for EPONs of 20 and 100 km. The graph shows that the algorithm guarantees the minimum bandwidth levels for every SLA, even when the distance increases up to 100 km. As a consequence, it can be stated that NN-SPID is not sensitive to high distances and can be applied in long-reach EPON architectures, achieving the same good performance.

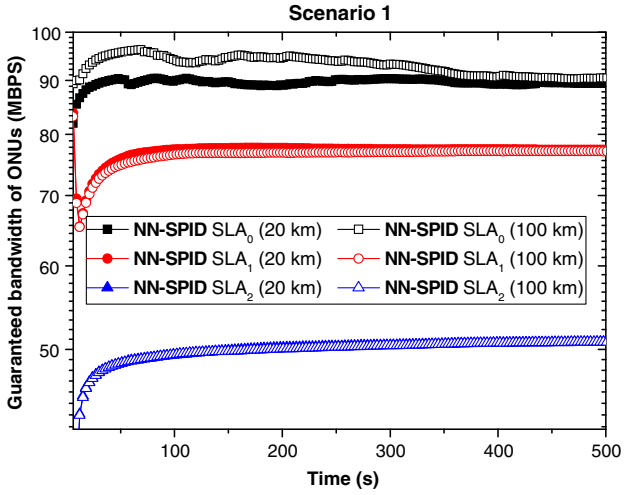

Fig. 12. Comparison of the real-time evolution of the mean allocated bandwidth of NN-SPID for different distances: 20 and 100 km.

IV. CONCLUSION

In this work, a PID control strategy to control QoS requirements based on an auto-adaptive neural network has been presented. The developed algorithm, NN-SPID, has been designed to control the bandwidth-allocation process so that different profiles are provided with minimum bandwidth levels according to their contracted SLA. The control of quality of service in access networks

Authorized licensed use limited to: SHIV NADAR UNIVERSITY. Downloaded on January 16,2023 at 11:41:12 UTC from IEEE Xplore. Restrictions apply.

has never been implemented, to our knowledge, using a neural network integrated in a PID.

The proposed algorithm has been compared with GA-SPID, a published algorithm designed for the same purpose as NN-SPID but using a genetic algorithm to automatically tune the PID controller. In this way, GA-SPID is an offline technique as it launches the genetic algorithm to calculate the best individuals (tuning parameters) for the controller and after that the PID starts the allocation process using these tuning parameters. In contrast, NN-SPID applies an online strategy to tune the controller, since the values of the tuning parameters are constantly evolving to the best ones based on the last network state. Furthermore, the designed neural network does not use a learning technique in which the network is provided *a priori* with a set of good examples to learn and work, but rather applies an incremental learning technique so it adaptively learns from the real-time network conditions, modifying the associated weights of every layer of the neural network—in contrast to traditional neural networks that keep these weights constant over time.

The results of the simulation study have demonstrated that NN-SPID ensures the minimum guaranteed bandwidth levels to every profile faster than GA-SPID and independently from the QoS requirements. In fact, depending on the network scenario, NN-SPID is able to reduce by up to ten times the required time to achieve the stipulated guaranteed bandwidth requirements. Furthermore, NN-SPID periodically adapts the tuning parameters, so it modifies the control signal of the PID to face real-time changes in the network or traffic conditions. On the contrary, GA-SPID does not adapt the tuning parameters and cannot efficiently react when changes in the network state happen. This performance was demonstrated when the guaranteed bandwidth levels were changed by service providers over time.

Our results exhibit how GA-SPID shows higher oscillations than NN-SPID in the bandwidth-allocation process, especially just after the change appeared in the network. As a consequence, it can be stated that NN-SPID shows more stability and robustness than GA-SPID in case service providers require real-time changes in the guaranteed bandwidth levels without interruptions in the offered services.

To our knowledge, the integration of PID controllers and neural networks has never been proposed to provide QoS in PONs. Hence, in this paper, we have developed an automatic and online PID tuning technique to efficiently ensure bandwidth requirements to different-priority profiles using an auto-adaptive neural network. One interesting future research focus is on controlling other QoS parameters using this technique, such as jitter, packet delay of sensitive traffic, or packet loss ratio. We are also working on adapting this auto-adaptive algorithm to provide QoS requirements in second-generation PONs (NG-PON2), called TWDM-PONs, which combine a more complex architecture based on a simultaneous time and wavelength division multiplexing [49,50].

Acknowledgment

This work has been funded by the Spanish Ministry of Science and Innovation (TEC2014-53071-C3-2-P and TEC2015-71932-REDT).

References

[1] "FTTH rises in the east," The Light Age, vol. 6, Feb. 2015 [Online]. Available: http://www.ftthcouncil.eu/documents/Publications/TLA6.pdf, accessed May 2016.

[2] D. Murayama, N. Oota, K. Suzuki, and N. Yoshimoto, "Low latency dynamic bandwidth allocation for 100 km long-reach EPONs," *J. Opt. Commun. Netw.*, vol. 5, pp. 48–55, Jan. 2013.

[3] J.-R. Lai and W.-P. Chen, "High utilization dynamic bandwidth allocation algorithm based on sorting report messages with additive-polling thresholds in EPONs," *Opt. Switching Netw.*, vol. 18, pp. 81–95, Nov. 2015.

[4] Ö. C. Turna, M. A. Aydin, A. H. Zaim, and T. Atmaca, "A new dynamic bandwidth allocation algorithm based on online–offline mode for EPON," *Opt. Switching Netw.*, vol. 152, pp. 29–43, June 2014.

[5] Y. Yin and G.-S. Poo, "User-oriented hierarchical bandwidth scheduling for Ethernet passive optical networks," *Comput. Commun.*, vol. 33, pp. 965–975, May 2010.

[6] B. Kantarci and H. Mouftah, "Two-stage report generation in long-reach EPON for enhanced delay performance," *Comput. Commun.*, vol. 36, pp. 1570–1580, Aug. 2013.

[7] S. Herrería-Alonso, M. Rodriguez Pérez, M. Fernández Veiga, and C. López García, "On the use of the doze mode to reduce power consumption in EPON system," *J. Lightwave Technol.*, vol. 32, pp. 285–292, Jan. 2014.

[8] I.-S. Hwang, Z.-D. Shyu, L.-Y. Ke, and C.-C. Chang, "A novel early DBA mechanism with prediction-based fair excessive bandwidth allocation scheme in EPON," *Comput. Commun.*, vol. 31, pp. 1814–1823, Apr. 2007.

[9] M. Mcgarry, M. Reisslein, and M. Maier, "Ethernet passive optical network architectures and dynamic bandwidth allocation algorithms," *IEEE Commun. Surv. Tutorials*, vol. 10, pp. 46–60, Oct. 2008.

[10] A. Nikoukar, I. Hwang, A. T. Liem, and C. Wang, "QoS-aware energy-efficient mechanism for sleeping mode ONUs in enhanced EPON," *Photon. Netw. Commun.*, vol. 30, pp. 59–70, Aug. 2015.

[11] A. Dixit, B. Lanoo, G. Das, D. Colle, M. Pickavet, and P. Demeester, "Dynamic bandwidth allocation with SLA awareness for QoS in Ethernet passive optical network," *J. Opt. Commun. Netw.*, vol. 5, pp. 240–253, Mar. 2013.

[12] T. Berisa, A. Bazant, and V. Mikac, "Bandwidth and delay guaranteed polling with adaptive cycle time (BDGPACT): A scheme for providing bandwidth and delay guarantees in passive optical networks," *J. Opt. Commun. Netw.*, vol. 8, pp. 337–345, Apr. 2009.

[13] M. Ma, Y. Zhu, and T. H. Cheng, "A bandwidth guaranteed polling MAC protocol for Ethernet passive optical networks," in *IEEE INFOCOM*, San Francisco, CA, 2003, pp. 22–31.

[14] F. T. An, H. Bae, Y. Hsueh, M. Rogge, L. Kazovsky, and K. Kim, "A new media access control protocol guaranteeing fairness among users in Ethernet-based passive optical networks," in *Optical Fiber Communication Conf.*, Atlanta, GA, 2003, pp. 134–135.

[15] C. Assi, Y. Ye, S. Sudhir, S. Dixit, and M. Ali, "Dynamic bandwidth allocation for quality-of-service over Ethernet PONs," *IEEE J. Sel. Areas Commun.*, vol. 21, pp. 1467–1477, Nov. 2003.

[16] C.-H. Chang, N. Merayo, P. Kourtessis, J. Senior, and R. M. Lorenzo, "Full-service MAC protocol for metro-reach GPONs," *J. Lightwave Technol.*, vol. 28, pp. 1016–1022, Apr. 2010.

[17] K. H. Ang, G. Chong, and Y. Li, "PID control system analysis, design and technology," *IEEE Trans. Control Syst. Technol.*, vol. 13, pp. 559–576, July 2005.

Authorized licensed use limited to: SHIV NADAR UNIVERSITY. Downloaded on January 16,2023 at 11:41:12 UTC from IEEE Xplore. Restrictions apply.

[18] K. J. Aström and T. Hägglund, *Advanced PID Control*. Research Triangle Park, NC: ISA—The Instrumentation, Systems, and Automation Society, 2006.

[19] M. I. Menhas, I. Wang, M. Fei, and H. Pan, "Comparative performance analysis of various binary coded PSO algorithms in multivariable PID controller design," *Expert Syst. Appl.*, vol. 39, pp. 4390–4401, Mar. 2012.

[20] V. Hultmann and L. dos Santos Coelho, "Tuning of PID controller based on a multiobjective genetic algorithm applied to a robotic manipulator," *Expert Syst. Appl.*, vol. 39, pp. 8968–8974, Apr. 2012.

[21] A. Bianciotto, B. J. Puttnam, B. Thomsen, and P. Bayvel, "Optimization of wavelength-locking loops for fast tunable laser stabilization in dynamic optical networks," *J. Lightwave Technol.*, vol. 27, pp. 2117–2124, June 2009.

[22] T. Tachibana, K. Kogiso, and K. Sugimoto, "Dynamic management of computing and network resources with PID control in optical grid networks," in *IEEE Int. Conf. on Communications (ICC)*, Beijing, China, 2008, pp. 396–400.

[23] T. Tachibana and K. Sugimoto, "Light-path establishment with PID control for effective wavelength utilization in all-optical wavelength-division multiplexing networks," *J. Opt. Commun. Netw.*, vol. 8, pp. 383–392, Dec. 2008.

[24] C. K. Chen, H. Kuo, and J. Yan, "GA-based PID active queue management control design for a class of TCP communication networks," *Expert Syst. Appl.*, vol. 36, pp. 1903–1913, Mar. 2009.

[25] T. Jiménez, N. Merayo, P. Fernández, R. J. Durán, I. de Miguel, R. M. Lorenzo, and E. J. Abril, "Implementation of a PID controller for the bandwidth assignment in long-reach PONs," *J. Opt. Commun. Netw.*, vol. 4, pp. 392–401, May 2012.

[26] T. Jiménez, N. Merayo, P. Fernández, R. J. Durán, I. de Miguel, R. M. Lorenzo, and E. J. Abril, "A PID-based algorithm to guarantee QoS delay requirements in LR-PONs," *Opt. Switching Netw.*, vol. 14, pp. 78–92, Aug. 2014.

[27] R. Rodriguez, A. Rodrigues, and L. dos Santos, "Computational intelligence approach to PID controller design using the universal model," *Inform. Sci.*, vol. 180, pp. 3980–3991, Oct. 2010.

[28] M. A. Keshari and A. Mehetab, "Study of the design and tuning methods of PID controller based on fuzzy logic and genetic algorithm," Bachelor of technology thesis, National Institute of Technology, Odisha, India, 2012.

[29] C. J. Lakhmi and N. M. Martin, *Fusion of Neural Networks, Fuzzy Sets, and Genetic Algorithms: Industrial Applications*, 1st ed. Boca Raton, FL: CRC Press, 1998.

[30] Y. Yuan, C. Chang, Z. Zhou, X. Huang, and Y. Xu, "Design of a single-input fuzzy PID controller based on genetic optimization scheme for DC-DC buck converter," in *Int. Symp. on Next-Generation Electronics (ISNE)*, Keelung, Taiwan, 2015, pp. 1–4.

[31] K. M. Hussain, R. A. Rajendran, M. S. Kumar, and S. M. Giriraj Kumar, "Comparison of PID controller tuning methods with genetic algorithm for FOPTD system," *Int. J. Eng. Res. Appl.*, vol. 4, pp. 308–314, Feb. 2014.

[32] H. Ali and S. Wadhwani, "Intelligent PID controller tuning for higher order process system," *Int. J. u- and e-Serv. Sci. Technol.*, vol. 8, pp. 323–330, 2015.

[33] M. J. Neath, A. K. Swain, U. K. Madawala, and D. J. Thrimawithana, "An optimal PID controller for a bidirectional inductive power transfer system using multiobjective genetic algorithm," *IEEE Trans. Power Electron.*, vol. 29, pp. 1523–1531, May 2013.

[34] V. Kozeny, "Genetic algorithms for credit scoring: Alternative fitness function performance comparison," *Expert Syst. Appl.*, vol. 42, pp. 2998–3004, Apr. 2015.

[35] M. Rajabi-Bahaabadi, A. Shariat-Mohayman, M. Babaei, and C. Wook, "Multi-objective path finding in stochastic time-dependent road networks using non-dominated sorting genetic algorithm," *Expert Syst. Appl.*, vol. 42, pp. 5056–5064, July 2015.

[36] T. Jiménez, N. Merayo, P. Fernández, R. J. Durán, I. de Miguel, R. M. Lorenzo, and E. J. Abril, "An auto-tuning PID control system based on genetic algorithms to provide delay guarantees in passive optical networks," *Expert Syst. Appl.*, vol. 42, pp. 9211–9220, Dec. 2015.

[37] H. Xu, J. Lai, Z. Yu, and J. Liu, "Based on neural network PID controller design and simulation," in *2nd Int. Conf. on Computer and Information Application (ICCIA)*, Taiyuan, China, 2012, pp. 514–517.

[38] L. Liu and J. Luo, "Research of PID control algorithm based on neural network," in *Proc. of Energy Procedia*, 2011, pp. 6988–6993.

[39] W. Lu, "The PID controller based on the artificial neural network and the differential evolution algorithm," *J. Comput.*, vol. 7, pp. 2368–2375, Oct. 2012.

[40] B. Yang, J. Chang, H. Su, J. Peng, S. Zhang, S. Guo, L. Zhang, C. Srinivasakannan, Z. Liu, Z. Li, and Z. Cao, "Self-adaptive PID controller integrated with RBFNN identification applied to microwave drying process," *J. Convergence Inform. Technol.*, vol. 8, pp. 779–783, Jan. 2013.

[41] S. Song and W. Liu, "Application of improved PID controller in motor drive system," in *PID Control, Implementation and Tuning*. Rijeka, Croatia: InTech, 2011.

[42] K. Vikas, G. Prerna, and A. P. Mittal, "ANN based self-tuned PID like adaptive controller design for high performance PMSM position control," *Expert Syst. Appl.*, vol. 41, pp. 7995–8002, Dec. 2014.

[43] J. Peng and R. Dubay, "Identification and adaptive neural network control of a DC motor system with dead-zone characteristics," *ISA Trans.*, vol. 50, pp. 588–598, July 2011.

[44] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Upper Saddle River, NJ: Pearson, 2009.

[45] K. J. Aström and T. Hägglund, *PID Controllers: Theory, Design and Tuning*, 2nd ed. Research Triangle Park, NC: ISA—The Instrumentation, Systems, and Automation Society, 1995.

[46] G. Kramer, B. Mukherjee, and G. Pesavento, "Interleaved polling with adaptive cycle time (IPACT): A dynamic bandwidth distribution scheme in an optical access network," *Photon. Netw. Commun.*, vol. 4, pp. 89–107, Jan. 2002.

[47] "OMNeT++ Discrete Event Simulator," 2015 [Online]. Available: https://omnetpp.org/.

[48] G. Kramer, "Synthetic self-similar traffic generation," 2001 [Online]. Available: http://glenkramer.com/trf_research.shtml.

[49] F. G. Effenberger, "PON resilience," *J. Opt. Commun. Netw.*, vol. 7, pp. A547–A552, Feb. 2015.

[50] H. Nakamura, "NG-PON2 technology," in *Optical Fiber Communication Conf. and the Nat. Fiber Optic Engineers Conf.*, Anaheim, CA, 2013, paper NTh4F.5.

Noemi Merayo received the telecommunication engineer degree in engineering from Valladolid University, Spain, in February of 2004 and the Ph.D. degree in the Optical Communication Group at the University of Valladolid, in July 2009. Since 2005, she has worked as a Junior Lecturer at the University of Valladolid.

Authorized licensed use limited to: SHIV NADAR UNIVERSITY. Downloaded on January 16,2023 at 11:41:12 UTC from IEEE Xplore. Restrictions apply.

She has also been a Visiting Research Fellow at the University of Hertfordshire (London), working in the Optical Networks Group, Science and Technology Research Institute (STRI). Recently, she has been a Postdoctoral Researcher in the group of "Transmisiones Ópticas de Banda Ancha (TOyBA)" of the University of Zaragoza. Her doctoral research focused on the design and performance evaluation of optical networks, especially passive optical networks.

D. Juárez studied telecommunication engineering at the University of Valladolid, Spain. He worked for the Optical Communications Group of the University of Valladolid from 2015 to 2016, where developed algorithms for resource management in PON and NGPON2 networks.

Juan Carlos Aguado received the telecommunication engineer and Ph.D. degrees from the University of Valladolid, Spain, in 1997 and 2005, respectively. He has worked as a Junior Lecturer at the University of Valladolid since 1998. His current research interests include the design and evaluation of cognitive methods applied to physical-layer modeling and traffic routing in heterogeneous optical networks.

I. de Miguel received his telecommunication engineer degree in 1997, and his Ph.D. degree in 2002, both from the University of Valladolid, Spain. Since 1997 he has worked as a Junior Lecturer at the University of Valladolid. He has also been a Visiting Research Fellow at University College London (UCL), working in the Optical Networks Group. His research interests include the design and performance evaluation of optical networks, especially hybrid optical networks, as well as IP over WDM. Dr. de Miguel is the recipient of the Nortel Networks Prize for the best Ph.D. Thesis on Optical Internet in 2002, awarded by the Spanish Institute and Association of Telecommunication Engineers (COIT/AEIT). He also received the 1997 Innovation and Development Regional Prize for his graduation project.

Ramón J. Durán was born in Cáceres, Spain, in 1978. He received his telecommunication engineer degree in 2002 and his Ph.D. degree in 2008, both from the University of Valladolid, Spain. Since 2002, he has worked as a Junior Lecturer at the University of Valladolid and currently is also Deputy Director of the Faculty of Telecommunication Engineering. His current research focuses on the design and performance evaluation of cognitive heterogeneous optical networks. Dr. Durán is the author of more than 60 papers in international journals and conferences.

P. Fernández obtained the telecommunication engineer degree from Universidad Politécnica de Cataluña, Barcelona, Spain, in 1997 and the Ph.D. degree in 2004 from University of Valladolid. Since 1999, she has worked as a Junior Lecturer at the University of Valladolid. Her research interests include passive optical networks and fiber-optic communications components. Dr. Fernández is the author of more than 40 papers in international journals and conferences. Currently, she is a Professor and Head of the Faculty of Telecommunication Engineering at the University of Valladolid.

R. M. Lorenzo received his telecommunication engineer and Ph.D. degrees from the University of Valladolid, Spain, in 1996 and 1999, respectively. From 1996 to 2000, he was a Junior Lecturer at the University of Valladolid, and joined the Optical Communications Group. Since 2000, he has been a Lecturer. His research interests include integrated optics, optical communication systems, and optical networks.

E. J. Abril received his telecommunication engineer and Ph.D. degrees from Universidad Politécnica de Madrid, Spain, in 1985 and 1987, respectively. From 1984 to 1986, he was a Research Assistant at Universidad Politécnica de Madrid, becoming Lecturer in 1987. Since 1995, he has been a Full Professor at the University of Valladolid, Spain, where he founded the Optical Communications Group. His research interests include integrated optics, optical communication systems, and optical networks. Prof. Abril is the author of more than 100 papers in international journals and conferences.

Authorized licensed use limited to: SHIV NADAR UNIVERSITY. Downloaded on January 16,2023 at 11:41:12 UTC from IEEE Xplore. Restrictions apply.