Intelligent Control and Automation, **2012, 3, 71-77**
http://dx.doi.org/10.4236/ica.2012.31009 Published Online February 2012 (http://www.SciRP.org/journal/ica)

Particle Swarm Optimization Based Fuzzy-Neural Like PID Controller for TCP/AQM Router

Mohammed Z. Al-Faiz, Shahad A. Sadeq
Computer Engineering Department, College of Engineering, Nahrain University, Baghdad, Iraq
Email: mzalfaiz@ieee.org, shahad.sadeq@yahoo.com

Received September 19, 2011; revised October 24, 2011; accepted November 2, 2011

ABSTRACT

In this paper a PID Fuzzy-Neural controller (FNC) is designed as an Active Queue Management (AQM) in internet routers to improve the performance of Fuzzy Proportional Integral (FPI) controller for congestion avoidance in computer networks. A combination of fuzzy logic and neural network can generate a fuzzy neural controller which in association with a neural network emulator can improve the output response of the controlled system. This combination uses the neural network training ability to adjust the membership functions of a PID like fuzzy neural controller. The goal of the controller is to force the controlled system to follow a reference model with required transient specifications of minimum overshoot, minimum rise time and minimum steady state error. The fuzzy membership functions were tuned using the propagated error between the plant outputs and the desired ones. To propagate the error from the plant outputs to the controller, a neural network is used as a channel to the error. This neural network uses the back propagation algorithm as a learning technique. Firstly the parameters of PID of Fuzzy-Neural controller are selected by trial and error method, but to get the best controller parameters the Particle Swarm Optimization (PSO) is used as an optimization method for tuning the PID parameters. From the obtained results, it is noted that the PID Fuzzy-Neural controller provides good tracking performance under different circumstances for congestion avoidance in computer networks.

Keywords: Neural Networks; Fuzzy Logic; PID Controller; AQM; PSO; Computer Network

1. Introduction

The term "congestion control" is used to describe the efforts made by network nodes to prevent or respond to overload conditions. Congestion in a computer network is a state in which performance degrades due to the saturation of network resources such as communication links, processor cycles, and memory buffers. Network congestion has been well recognized as a resource-sharing problem. When too many packets are difficult to be described as a mathematical model, and the controller can be designed to apply heuristic rules contending for the same link, the queue overflows and packets have to be dropped.

When such drops become common events, the network is said to be congested. Most networks provide a congestion-control mechanism to deal with just such a situation [1]. The Internet Engineering Task Force (IETF) has proposed the deployment of active queue management (AQM) mechanisms [2] at gateways to improve the performance of TCP congestion control. AQM has been a very active research area in the Internet community. Random early detection (RED) [3] is an extensively studied AQM algorithm that can detect congestion. It controls congestion by randomly dropping packets with certain probability that is a function of the average queue size (q_{avg}). In recent years, the more needs for the congestion controllers having enough ability which is more logically predictable and reliable are occurred. For this reason, the traditional control algorithms which have been used only for mechanical or electrical systems are adopted to the area of congested network and their performances which are known as relatively good. Consequently, the more controllers which have various featured types have been adopted to the network congestion control area using control theories. On the issue of applying control theories to the network, especially AQM Router, a proportional (P) controller and a proportional-plus-integral (PI) controller for AQM [4] were designed based on the classical control theory and the dynamic model of the TCP congestion control [5] and its liearized model [6]. And also, [7] proposed an adaptive fuzzy AQM for congestion avoidance in TCP/AQM networks. More recently [8] developed a new AQM algorithm based on neural networks. The proposed controller is simple and can be easily implemented in high-speed routers.

In [9] proposed the adoption of a Fuzzy Proportional Integral (FPI) base genetic controller as an AQM for internet router.

Here, a Fuzzy-Neural like PID controller based PSO is designed as an AQM for internet router.

Developments in intelligent control had been made in the past years through the development of fuzzy logic control (FLC) and Neural Networks (NN) [10].

The main advantage of the Fuzzy Logic Controller (FLC) is that it can be applied to plants that are that reflect the experience of human experts. PID FLCs have been successfully applied to a variety of practical problems. In spite of its practical success, there is no standard procedure for tuning PID FLCs [11].

Artificial Neural Network (ANN) is a combination of processing elements that perform certain tasks through learning weights.

Neural Networks provide a different approach to problem solving from linguistic or algorithmic systems such as FLC.

By combining both algorithms of NNs and FLCs together a robust controller may be achieved, which can give precise actions and learn to enhance its performance. A FLC can represent human reasoning while NN can simulate human learning [12].

The organization of the paper is as follows: Section 2 describes the linearized AQM model. Section 3 describes the Fuzzy Neural Network (FNN) design.

The simulation results are given in Section 4 to verify the proposed controller using MATLAB package. Finally, a conclusion is given in section 5.

2. TCP/AQM System Model

AQM has been extensively analyzed using control-theoretical methods. Control-theoretical approaches lead to stable, effective, and robust congestion control operation. In [5], the non-linear dynamic model for multiple TCP flows control has been developed based on fluid-flow theory to model the interactions of a set of TCP flows and AQM routers in computer networks which consist of a system of nonlinear differential equations. For the control theoretical analysis, it was approximated as a linearized constant model by small signal linearization about an operating point (W_0, q_0, p_0), see [6] for linearization details, which leads to the following :

$$\delta\dot{W}(t) = -\frac{2N}{R_0^2 C}\delta W(t) - \frac{R_0 C^2}{2N^2}\delta p(t - R_0) \quad (1)$$

$$\delta\dot{q}(t) = \frac{N}{R_0}\delta W(t) - \frac{1}{R_0}\delta q(t) \quad (2)$$

where $\delta W(t) \approx W - W_0$, $\delta q(t) \approx q - q_0$, $\delta p(t) \approx p - p_0$, $\dot{W}(t)$ denotes the time-derivative of $W(t)$, $\dot{q}(t)$ denotes the time derivativeof $q(t)$, and

W: Expected TCP window size (packets);

q: Expected queue length (packets);

R_0: Round-trip time (seconds);

C: Link capacity (packets/second);

N: Load factor (number of TCP sessions);

p: Probability of packet mark/drop;

t: Time.

The expected queue length q and the expected TCP window size W are positive value and bounded quantities. And also, the probability of packet (mark/drop) p takes value only in [0,1].

Taking the Laplace transform of Equation (1) and rearranging the following transfer functions are obtained:

$$P_{\mathrm{TCP}}(s) = \frac{W(s)}{p(s)} = \frac{\frac{R_0 C^2}{2N^2}}{s + \frac{2N}{R_0^2 C}} \quad (3)$$

$$P_{\mathrm{queue}}(s) = \frac{q(s)}{W(s)} = \frac{\frac{N}{R_0}}{s + \frac{1}{R_0}} \quad (4)$$

So, the overall plant transfer function becomes:

$$P(s) = P_{\mathrm{TCP}}(s) P_{\mathrm{queue}}(s) e^{-sR_0} \quad (5)$$

And can be expressed as:

$$P(s) = \frac{\delta q(s)}{\delta p(s)} = \frac{\frac{C^2}{2N} e^{-sR_0}}{\left(s + \frac{2N}{R_0^2 C}\right)\left(s + \frac{1}{R_0}\right)} \quad (6)$$

Thus, the block diagram of linearized AQM control system is shown in **Figure 1**. In this diagram $P_{\mathrm{TCP}}(s)$ denotes the transfer function from loss probability $\delta p(t)$ to window size $\delta W(t)$, $P_{\mathrm{queue}}(s)$ denotes the transfer function from $\delta W(t)$ to queue length $\delta q(t)$, and $C(s)$ denotes the transfer function of controller. Taking the Z-transform to Equation (6), the designed plant transfer function is obtained after considering the sampling time half of R_0. Precisely and for consider the case study with $N = 60$, $C = 3750$ packets/sec and $R_0 = 0.253$ sec the following discrete transfer function are obtained.

$$P(z) = \frac{q(z)}{p(z)} = \frac{11252.46 z^{-3}}{1 - 1.545 z^{-1} + 0.569 z^{-2}} \quad (7)$$

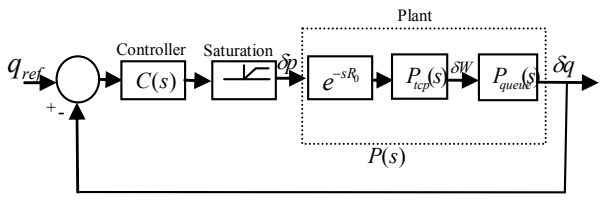

Figure 1. Block diagram of a linearized AQM as feedback control.

3. Fuzzy Neural Network Design

In order to design a FNC a Neural Network (NN) should be designed first. This NN represents the structure of FLC. It's called Fuzzy Neural Network (FNN) Structure. In this work, it assumed that a Mamdani type FLC with two inputs of error and rate of error and one output is used. The memberships used for the inputs and output are bill shaped type with 7 memberships for each. These rules are reduced from 7 × 7 to 7 only since any input has some contribution in all of the fuzzy sets and will circle around the main diagonal of the fuzzy rule table and settle in the center of this table. Hence, fuzzy rule table will be as shown in **Table 1**.

where the abbreviations of the table represent: PB as Positive Big, PM as a Positive Medium, PS as Positive Small, Z as Zero, NS as Negative Small, NM as Negative Medium and NB as Negative Big. Moreover, the rules are implied using product for AND operation. Furthermore, the defuzzification used in this controller is a center of gravity type.

3.1. FNN Structure

The structure of FNN is shown in **Figure 2**. This structure consists of five layers which are:

1) Input layer: in the input layer, each node transmits the corresponding input to the antecedent layer, thus:

$$X_i^1 = I_i \tag{8}$$

and

$$O_i^1 = X_i^1 \tag{9}$$

where

I_i is the i^{th} network input,

X_i^1 is the node input, and

I_i is the node output.

The subscript refers to the node number while the superscript refers to the layer number.

2) Antecedent layer: this layer will transmits each value of input to the corresponding linguistic set, hence:

$$X_i^2 = -\frac{1}{2}\cdot\left(\frac{O_j^1 - m_{ij}}{s_{ij}}\right)^2 \tag{10}$$

and

$$O_i^2 = e^{\left(X_i^2\right)} \tag{11}$$

where i = 1,2 ⋯ 14, j = 1,2,⋯, m_{ij} is the center of bill shaped fuzzy membership function, s_{ij} is the standard deviation, i refers to the antecedent node while j refers to the input node.

3) Rule layer: this layer performs the implication of AND operation. The rule is implied using product operation. Since only 7 rules will contribute the rules which represent the diagonal of the fuzzy rule table mentioned in **Table 1**, then:

$$X_i^3 = O_i^2 \cdot O_{i+7}^2 \tag{12}$$

and

$$O_i^3 = X_i^3 \tag{13}$$

where i = 1,2 ⋯ 7.

4) Consequent layer: only two nodes are available in this layer which performs the center of gravity defuzzification algorithm. The first node has a weighted input and the second is of the strength of unity, thus:

$$X_1^4 = \sum_{i=1}^{n} O_i^3 \cdot y_i \tag{14}$$

$$X_2^4 = \sum_{i=1}^{n} O_i^3 \tag{15}$$

and

$$O_i^4 = X_i^4 \tag{16}$$

where i = 1,2. n is the number of rules y_i is the weight between the rule and the consequent layers.

5) Action layer: the completion of center of gravity defuzzificztion algorithm is done in this layer, so the input and output of each node is given by:

$$X^5 = \frac{O_1^4}{O_2^4} \tag{17}$$

and

$$O^5 = X^5 \tag{18}$$

Table 1. Rules of fuzzy logic controller.

Change Of Error (ce) \ Error (e)	NB	NM	NS	Z	PS	PM	PB
NB	PB						
NM		PM					
NS			PS				
Z				Z			
PS					NS		
PM						NM	
PB							NB

3.2. FNN Learning Algorithm

FNN structure is expressed analytically in the previous section, such that the optimization method (Steepest Descent) can be applied on such structure. Hence, it can be implemented in the layers as follows:

1) Consequent layer (layer four):

$$y_i(k+1) = y_i(k) + \frac{\eta \cdot O_i^3 \cdot \left(O_d(k) - O^5(k)\right)}{O_2^4} \quad (19)$$

where η is the learning rate, O_d is the desired output.

2) Antecedent layer (layer two): In this layer, the learning equation of back propagation is:

$$m_{ij} = (k+1) = m(k) + \eta \cdot O_i^3 \cdot \left(O_d(k) - O^5(k)\right) \cdot \left(y_i(k) - O^5(k)\right) \cdot \frac{\left(O_j^1 - m_{ij}\right)}{O_2^4 \cdot \left(s_{ij}\right)^2} \quad (20)$$

and

$$s_{ij}(k+1) = s_{ij}(k) + \eta \cdot \left(O_d(k) - O^5(k)\right) \cdot \left(y_i(k) - O^5(k)\right) \cdot \frac{\left(O_j^1 - m_{ij}\right)^2}{O_4^2 \cdot \left(s_{ij}\right)^3} \quad (21)$$

Equations (19), (20), and (21) will perform the back propagation procedure. Initializing the parameters in FNN is very important since it may reduce the learning time of the network, thus these parameters have been chosen at the same bases of choosing them in an ordinary fuzzy logic controller. That is m_{ij} will divide the universe of discourse to 7 equal intervals, while s_{ij} will give the bill shaped functions a reasonable width. Finally, y_i is scattered along the output universe of discourse in an equal intervals.

3.3. PID-Like Fuzzy Neural Controller (PID-FNC)

The equation of PID controller in time domain is:

$$u(t) = K_P \cdot e(t) + K_D \cdot \delta e(t) + K_I \cdot \int e(t) \cdot \mathrm{d}t \quad (22)$$

where K_P, K_I and K_D are the proportional, integral and derivative gains of the PID respectively.

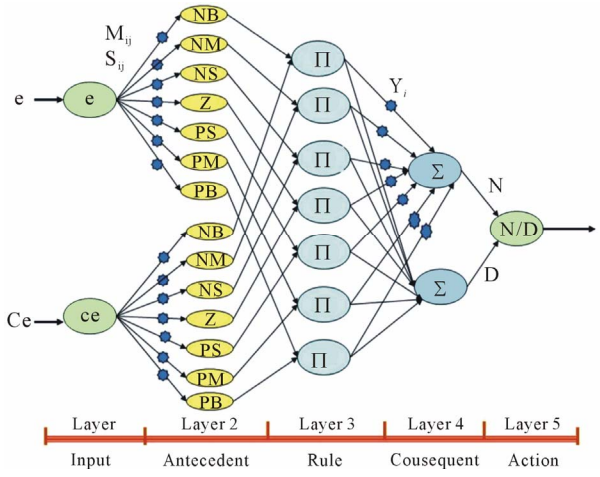

Figure 2. Fuzzy neural network structure (FNN).

Thus, in the discrete case of a PID like fuzzy controller one has an additional process state variable, namely sum-of-error to dente the integral part. Unfortunately, if any input is described with (m) linguistic value, then since PID controller has three inputs and since any rule has three conditions, then there is a need of $m \times m \times m = m^3$ rules. So, it is too much work to write m^3 rules. The PID-like fuzzy controller can be constructed as a parallel structure of a PD-like fuzzy controller and a PI-like fuzzy controller and the output can be approximated as [13]:

$$u(t) = u_a + u_b = \left(K_{P1} \cdot e + K_D \cdot \delta e(t)\right) + \left(K_{P2} \cdot e + K_I \cdot \int e \cdot \mathrm{d}t\right) \quad (23)$$

However, the first part of Equation (23) represents PD controller. PD controller for any pair of the values of e and δe, calculates the control signal (u_a).

$$u_a(t) = K_{P1} \cdot e(t) + K_D \cdot \delta e(t) \quad (24)$$

The fuzzy controller should do the same thing. For any pair of error (e) and change of error (δe), it should work out the control signal though rules. In fuzzy rules, the sampling time will be omitted since such a rule expresses a causal relationship between the process state and control output variables, which holds for any sampling time.

Moreover, the second part of Equation (23) represents PI controller, which can represent PI like fuzzy controller. The Fuzzy controller and the rules table have other inputs; error and sum of error. It means that, the rules themselves should be reformulated. Since only the diagonal of the rule table will be used, it is found that the rules of PI-controller part are the same rules mentioned in **Table 1**. However, the equation of PI controller is:

$$u_b(t) = K_{P2} \cdot e(t) + K_I \cdot \int e(t) \cdot \mathrm{d}t \quad (25)$$

The proposed PID-FNC will consist of two FNNs. The first one is for PD like controller action mentioned in Equation (23), while the second FNN is for PI like controller action mentioned in Equation (25). The general block diagram of PID-FNC is shown in **Figure 3**. The parameters of K_{ua} and K_{ub} are the output scaling factors of PD like and PI like fuzzy controllers respectively. It will be assumed in this work that there is no need to any rule definition, since the rule layer is fixed and take the optimal rules of the fuzzy logic controller. However, the general Block Diagram of the PID-FNC Controlled System is shown in **Figure 4**. In this figure, Neural Network Emulator (NNE) is used to emulate the plant model. Hence, it is used to generate the required propagated error signal to PID-FNC (FNCe). This NNE uses the back propagation algorithm as a learning technique and uses the error between the plant and NNE, (NNEe), as the learning signal to adjust its weights. PID-FNC uses the

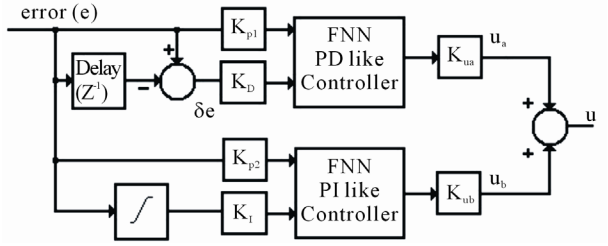

Figure 3. General block diagram of PID-FNC.

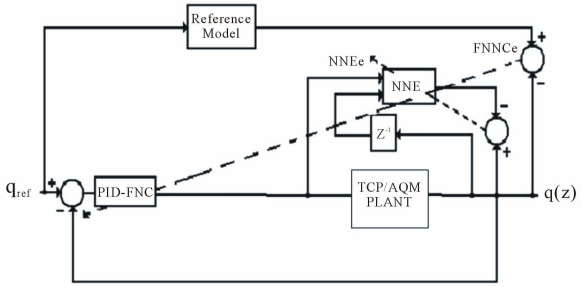

Figure 4. General block block diagram of PID-FNC controlled system.

error between the plant output and the reference model output to generate both error, sum of error and change of error internally. Then after generating the controller action, it updates its weights using FNCe; which is the error between the reference model output and the plant output propagated through NNE.

4. Simulation

In this section the proposed controller is evaluated using MATLAB.

Figure 5 shows the network topology used for this set of simulation in which the shared bottleneck link between router R_1 and router R_2 has a capacity of 15 Mbps with propagation delay of 20 ms and N = 60 and the packet size is set to be 500 bytes and the reference input (queue size) which has rectangular form changes every 50 seconds as shown in Equation (26).

$$q_{ref} = \begin{cases} 300 & 0 < t < 50; \\ 200 & 50 < t < 100; \\ 400 & 100 < t < 150; \\ 200 & 150 < t < 200; \end{cases} \quad (26)$$

The maximum queue length in the AQM router Router 1 is 800 packets. The AQM mechanism is configured at Router1, and drop Tail is used at other gateways

First the simulation is done for the system without controller as shown **Figure 6**.

Figure 6 shows that the system without controller is unable to track the queue length around the queue length to the desired level, where the system goes into a sustained oscillation with high congestion exceeding the

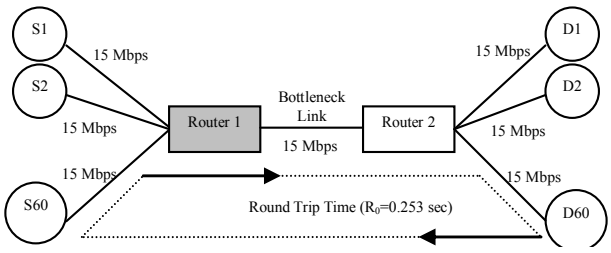

Figure 5. Network topology case study.

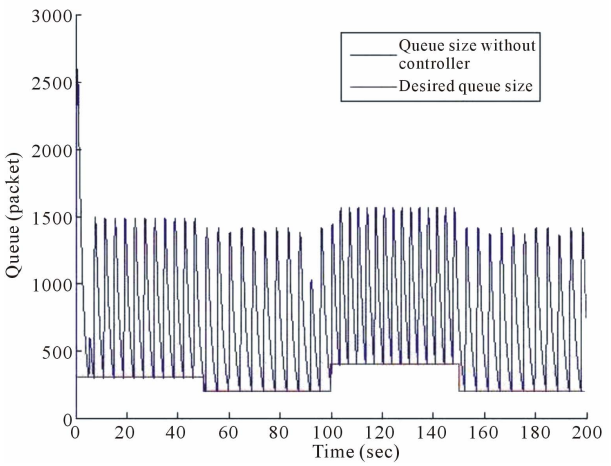

Figure 6. System response without controller.

maximum buffer size. In [9] a classical control (PI-controller) is applied in order to eliminate this sustained oscillation and get better tracking performance, also FPI controller is designed to speed up the system response, and FPI based genetic is designed to get the best parameters of FPI controller and to enhance the system response.

In this paper we used the PSO as a suitable optimization method for tuning FPI Parameters, the PSO parameter was: population size 80, the inertia weight factor w is 0.9, acceleration constant c_1 = 1.2 and c_2 = 0.12 and the fitness function is the integral time absolute error

$$\text{ITAE} = \sum_{0}^{T} T\left(q_{ref} - q\right)$$

The FPI parameters obtained in PSO are ($K_p = 1\times10^{-3}$, $K_i = 1\times10^{-3}$, $K_u = 5\times10^{-3}$).

It is found that that the FPI baesd PSO is better than FPI and could speed up the system response with less computation time than FPI based G.A as shown in **Figure 7**.

Although the FPI based PS0 shows good performance the system response has overshoot and to overcome this drawback, first we design a Fuzzy-Neural like PI (FNPI) controller to can compare it with FPI and show how can the addition of NN can improve the system response.

To design a FNPI controller, we use the second part

only of **Figure 3**.

The FNPI controller parameters are selected first by trial and error method as follows (K_{p2} = 200, K_i = 7761200, K_u = 3×10^{-3}) and then by using PSO method to find the best parameters of FNPI, we get (K_{p2} = 300, K_i = 8861200, K_{ub} = 2×10^{-3}). As shown in **Figures 8** and **9**.

From **Figure 9** it was found the FNPI based on PSO is better than FPI based PSO by decreasing the overshoot of the system response.

Finally we design a Fuzzy Neural Controller like PID (FNC-PID) and show how it can improve the system response more than the FNPI.

The FNC-PID parameters are selected first by trial and error method as follows (K_{p1} = 1, K_d = 50.1261, K_{ua} = 1×10^{-5}, K_{p2} = 80.2745, K_i = 50.2638, K_{ub} = 2×10^{-3}). and then by using PSO method to find the best parameters of FNC-PID, we get (K_{p1} = 100.1616, K_d = 20.1261, K_{ua} = 1×10^{-5}, K_{p2} = 140.5, K_i = 30.2638, K_{ub} = 2×10^{-3}) as shown in **Figures 10** and **11**.

Figure 11 shows that FNC-PID based PSO is best than FNPI based PSO controller by making the response of the system more faster with zero overshoot.

The overall simulation results are shown in **Table 2**.

5. Conclusions

From the design and the simulation results, it can be concluded that:

1) The designed FPI based PSO is better than FPI which can improve the system response by decreasing it's overshoot and make it more faster by decreasing it's settling time which proves the efficiency of PSO as a suitable optimization method.

2) By using the PSO to optimally select the best pa

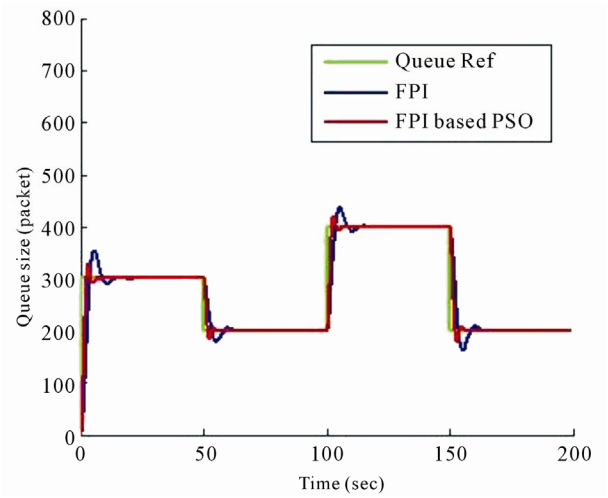

Figure 7. System response with FPI and FPI based PSO controller.

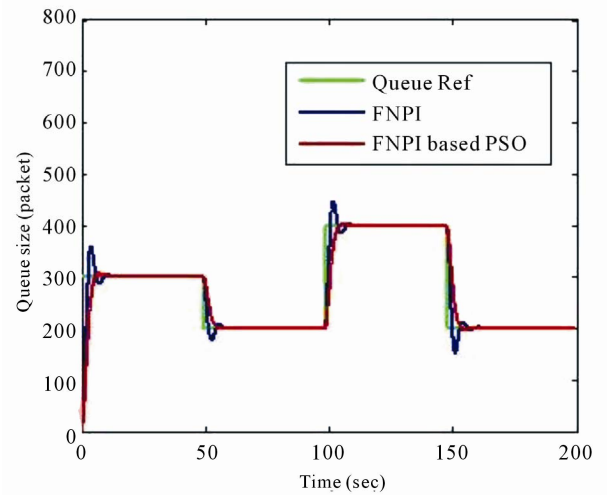

Figure 8. System response with FNPI, FNPI based PSO controller.

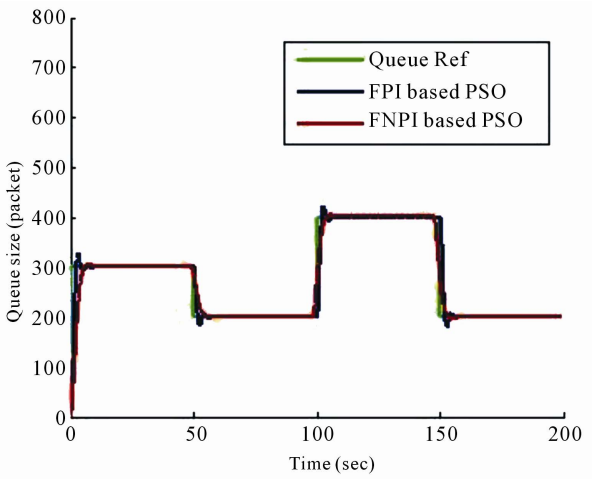

Figure 9. System response with FPI based PSO, FNPI based PSO controller.

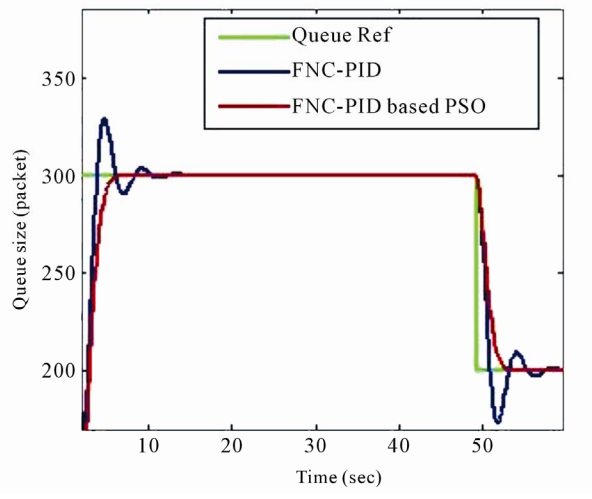

Figure 10. System response with FNC-PID, FNC-PID based PSO controller.

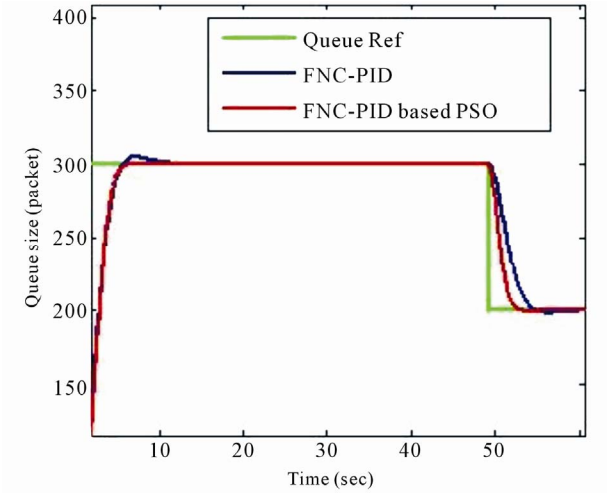

Figure 11. System response with FNPI based PSO, FNC-PID based PSO controller.

Table 2. TCP/QAM system response performance of FPI, FNPI, and FNC-PID based PSO with error criteria 5%.

Controller	Rise Time $(M_p\%)$ (sec)	Overshoot $M_p\%$	Peak Time (t_p) (sec)	Settling Time (t_p) (sec)
FPI	4	54	5	8
FPI-PSO	2.5	28	3	4
FNPI	2.7	60	3.8	12
FNPI-PSO	5.7	5	7	13
FNC-PID	3.8	29	4.7	15
FNC-PID-PSO	6.5	-	-	7

rameters of FNPI, the system response is improved by decreasing its overshoot as shown in **Table 2** which also proves the efficiency of PSO as a suitable optimization method.

3) The designed FNPI based PSO controller can decrease the overshoot of system response comparing with FPI based PSO controller.

4) The designed FNC-PID based PSO is better than FNC-PID controller which can improve the system response by making its overshoot zero and faster by decreasing its settling time which also proves the efficiency of the PSO as a suitable optimization method.

5) The designed FNC-PID based PSO controller can decrease the overshoot of system response of FNPI based PSO controller by making it zero and make the system response faster by decreasing its settling time.

6) The designed FNC-PID based PSO controller can deal with congestion problem with a good tracking performance about the desired queue size with high link utilization and faster system response.

REFERENCES

[1] L. Peterson and B. Davie, "Computer Network a Systems Approach," 3rd Edition, Morgan Kaufmann, Waltham, 2003.

[2] B. Braden, D. Clark and J. Crowcroft, "Recommendations on Queue Management and Congestion Avoidance in the Internet," Network Working Group, Internet Society, Reston, 1998.

[3] S. Floyd and V. Jacobson, "Random Early Detection Gateway for Congestion Avoidance," *IEEE Transactions on Networking*, Vol. 1, No. 4, 1993, pp. 397-413. doi:10.1109/90.251892

[4] C. V. Hollot, V. Misra, D. Towsley and W. B. Gong, "On Designing Improved Controllers for AQM Routers Supporting TCP Flows," *Proceedings of IEEE INFOCOM*, Anchorage, 24-26 April 2001, pp. 1726-1734.

[5] V. Misra, W.-B. Gong and D. Towsley, "Fluid-Based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED," *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, Stockholm, 28 August-1 September 2000, pp. 151-160.

[6] C. V. Hollot, V. Misra, D. Towsley and W. B. Gong, "A Control Theoretic Analysis of RED," *Proceedings of IEEE INFOCOM*, Anchorage, 24-26 April 2001, pp. 1510-1519.

[7] M. Jalili, F. Roudsari, A. Dehestani and H. Fesharaki "Adaptive Fuzzy Active Queue Management," *Proceedings of FORTE Workshops*, Islamic Azad University, Tehran, 2004, pp. 196-208.

[8] M. Y. Waskasi, M. J. Yazdanpanah and N. Yazdani, "A New Active Queue Management Algorithm Based on Neural Networks PI," University of Tehran, Tehran, 2005.

[9] M. Z. AL-Faiz and A. M. Mahmood, "Fuzzy Genetic Controller for Congestion Avoidance in Computer Networks," *Proceeding of Engineering Conference on Control, Computer and Mechatronics*, Baghdad, 30-31 January 2011, pp. 206-212.

[10] B. Kosko, "Neural Networks and Fuzzy Systems," Prentice Hall, Upper Saddle River, 1992.

[11] D. T. Pham, "Neural Networks for Identification, Prediction and Control," Springer, Berlin, 1995. doi:10.1007/978-1-4471-3244-8

[12] M. Brown, and C. Harris, "Neuro-Fuzzy Adaptive Modeling and Control," Prentice-Hall Inc., Englewood Cliffs, 1994.

[13] L. Reznik, "Fuzzy Controllers," Biddles Ltd., Norfolk, 1997.

