Research Article

Research on Virtual Machine Response Time Prediction Method Based on GA-BP Neural Network

Jun Guo, Shu Liu, Bin Zhang, and Yongming Yan

College of Information Science & Technology Engineering, Northeastern University, Shenyang, China

Correspondence should be addressed to Shu Liu; liushu_neu.edu@163.com

Received 17 March 2014; Revised 18 May 2014; Accepted 1 June 2014; Published 17 June 2014

Academic Editor: Qingsong Xu

Copyright © 2014 Jun Guo et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Cloud application provides access to large pool of virtual machines for building high-quality applications to satisfy customers' requirements. A difficult issue is how to predict virtual machine response time because it determines when we could adjust dynamic scalable virtual machines. To address the critical issue, this paper proposes a prediction virtual machine response time method which is based on genetic algorithm-back propagation (GA-BP) neural network. First of all, we predict component response time by the past virtual machine component usage experience data: the number of concurrent requests and response time. Then, we could predict virtual machines service response time. The results of large-scale experiments show the effectiveness and feasibility of our method.

1. Introduction

A critical issue that must be solved for the cloud application providers is to allocate and manage virtual resources because virtual machines are dynamic scalability in the cloud environment [1]. Cloud application customers expect to receive a certain level service performance as specified in SLA (service level agreement) [2]. Response time is an important indicator in the SLA. Providers dynamically adjust resources by checking whether the service response time exceeds the threshold ruled by SLA. However, it may lead to hysteresis and cause frequent adjustment of resources. Dynamic resource allocation scheme adjusts resources when multiple predicted response time values exceed the threshold of SLA. Therefore, predicting service component response time has high reference value for determining the right time to adjust resources. Traditional service component response time prediction methods do not consider the dynamic scalability of cloud environment [3]; therefore, they are not applicable to service component response time prediction in the cloud environment. It becomes an urgent task to explore an effective virtual machine service component response time prediction method.

In this paper, we adopt GA-BP neural network model to predict the virtual machine component response time. It

needs not consider the relationship between component concurrent request and response time or mathematical model. The model only requires training to establish the mapping relationship on the network, similar to "black box" method. The BP neural network has several advantages: nonlinear mapping ability, reducing the error between the predicted values and real value, and increasing speed of system processing [4]. GA-BP neural network model overcomes BP algorithm's problem, such as slow convergence and local optimal solution, and solves genetic algorithm's problem of approximate optimal in a short period. Thus the model has not only the self-learning ability of neural network but also the global searching ability of genetic algorithm.

In this paper, we propose a virtual machine service component response time prediction model based on GA-BP neural network. We collect three kinds of data: the number of components concurrent requests, initialized configuration parameters of virtual machine, and component response time. Then, we predict the total number of components concurrent requests according to the weight of every virtual machine. Finally, we predict service component response time on the virtual machine.

The experiments indicate that GA-BP network model is more effective than traditional BP network model, and that service component response time prediction algorithm based

FIGURE 1: Cloud application architecture.

on GA-BP network is more accurate than typical algorithms. The algorithm could be used as a reference for time selection in resource adjustment decision.

The rest of the paper is organized as follows. Section 2 gives a brief overview of existing research related to service response time model and BP neural network. Section 3 presents component response time prediction algorithm in detail. Section 4 shows the experimental results and Section 5 concludes the paper.

2. Related Work

At present, the research on service response time prediction is mainly in the field of Web service. In [5], it uses time series achieved good service response time prediction results in the field of Web services and grid computing. However, it does not apply to cloud application because cloud application is dynamic scalability and the number of virtual machines and resources is constantly changing [3]. It will break the development law of service response time. What is more, the method needs to build model of response time and it will degrade performance.

In [6], it uses queuing theory to predict response time. Queuing theory is a mathematics theory to study stochastic service system [7]. In [8], it builds the target system model, simplified system, and abstract system. It assumes that the arrival time obeys Poisson process and service time obeys exponential distribution. In fact, the real environment does not completely obey assumption.

BP neural network is a common method of training artificial neural networks to predict service response time [4]. In [9], it uses BP neural network to predict time delay. The network learns from many inputs for a desired output. It is effective with nonlinear approximation, but the convergence of the algorithm is very slow and may have local minimum problem. Genetic algorithm is a search heuristic that mimics the process of natural selection [10]. It overcomes BP algorithm's problem of slow speed of convergence and local minimum.

Previous studies always used genetic algorithm and BP neural network to predict traffic and futures, but it was not used to predict virtual machine service response time in cloud. Therefore, the paper proposed response time prediction method based on GA-BP neural network. It could overcome the shortages of previous work which could not adapt to resource dynamic scalability and accuracy. Moreover, it could satisfy the service performance as specified in SLA.

3. Virtual Machine Response Time Prediction Model

3.1. Cloud Application Architecture. Figure 1 shows the cloud application architecture. Service can be implemented by combing several virtual machines in the cloud application. The same set of components is deployed on several virtual machines, and balanced workload virtual resource pool is combined with several virtual machines. QOS will be affected by performances of other components on the virtual machine and performances of other virtual machines on the cloud application.

3.2. Response Time Prediction Model. In this section, we propose a virtual machine service component response time prediction model which is based on GA-BP neural network to solve random volatility of concurrent requests and virtual machine dynamic scalability. Figure 2 shows the response time prediction produce. First of all, we predict every component response time on the virtual machine. Then we predict virtual machine service component total response time.

Virtual machine component response time prediction process is divided into two situations.

- (1) The collected data could predict response time of the old virtual machine (it has been used before).
- (2) The collected data could not predict response time of the new virtual machine (it has been not used before).

There are three steps to predict the new virtual machine component response time.

FIGURE 2: VM response time prediction model.

Step 1. Collect the number of components concurrent requests and components response time.

Step 2. Use the history data to find the old virtual machine which is the most similar to the new virtual machine.

Step 3. The number of component concurrent requests is put into prediction model. Calculate the new virtual machine component response time.

The prediction method of the old virtual machine is similar to the new virtual machine. Because the old virtual machine has a lot of history data to predict component response time; we omit the third step. After predicting component response time in the next period, we calculate

service response time by weighting every virtual machine component response time.

4. Component Response Time Prediction Model

In this section, we propose a virtual machine component response time prediction model based on GA-BP neural network. In Section 4.1, we use similarity calculation model to find out the most similar virtual machine and use the most similar virtual machine to find out the relationship between component current requests and component response time. In Section 4.2, we determine the structure of BP neural network, critical parameters, and genetic operator.

In Section 4.3, we set up genetic algorithm parameters to predict accurate component response time. In Section 4.4, service component response time is calculated.

4.1. Virtual Machine Configuration Similarity Computation Model. There are n old virtual machines. If we add a new virtual machine, the configuration data will be put into the database. We choose m performance parameters to calculate similarity. The set of n old virtual machines is denoted as $X = \{x_1, x_2, \dots, x_n\}$. Performance parameters of virtual machine x_1 are denoted as $x_i = \{x_{i1}, x_{i2}, \dots, x_{im}\}$, $i = 1, 2, \dots, n$. The original virtual machine characteristic matrix is denoted as follows:

$$\begin{matrix} \text{VM}_1 \\ \text{VM}_2 \\ \vdots \\ \text{VM}_n \end{matrix} \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix}. \quad (1)$$

Performance parameters include CPU speed, memory capacity, I/O rate, and network bandwidth. We normalize virtual machine performance parameter configuration matrix and use weighted Euclidean distance to calculate two virtual machines similarity. Computation formula is denoted as follows:

$$d_{ij} = \|x_i - x_j\|_w = \sqrt{(x_i - x_j) W^T W (x_i - x_j)^T}; \quad (2)$$

$W = \{w_1, w_2, \dots, w_m\}$ $\sum_{i=1}^m w_i = 1$ is weight of m performance parameters set.

We calculate similarity to find the most similar virtual machine. We need a lot of data to train the relationship between the number of virtual machine component current requests and component response time.

4.2. BP Neural Network Construction. There are n components deployed on the virtual machine. There are N groups sample. The relationship between component current requests and component response time is $\{T(c_1), T(c_2), \dots, T(c_n)\} = F\{N(c_1), N(c_2), \dots, N(c_n)\}$.

The algorithm is multiple inputs and outputs. The input is the number of components current requests and the output is component response time. Hidden layer has P neurons and the input is $A_k = (N(c_1), N(c_2), \dots, N(c_n))$, $k = 1, 2, \dots, N$. The weight from input layer to hidden layer is w_{ij} . Hidden layer threshold is θ_j ; hidden layer weight input is $S_k = (s_1, s_2, \dots, s_p)$; and weight output is $B_k = (b_1, b_2, \dots, b_p)$. The weight from hidden layer to input layer is v_{jt} ; output layer threshold is r_t ; and output layer weighted value input is $B_k = (b_1, b_2, \dots, b_p)$. Expected response time output is $Y_k = (T(c_1), T(c_2), \dots, T(c_n))$ $\hat{Y}_k = (\hat{T}(c_1), \hat{T}(c_2), \dots, \hat{T}(c_n))$ $i = 1, 2, \dots, n$; $j = 1, 2, \dots, p$; $t = 1, 2, \dots, N$. Hidden layer activation function is $f(x)$; input of hidden layer is $s_j = \sum_{i=1}^n w_{ij} N(c_i) + \theta_j$; and output is $b_j = f(s_j)$, respectively. Output layer activation function is $g(x)$. Input of output layer is $l_t = \sum_{j=1}^p v_{jt} b_j + r_t$, and output of BP neural network is

$$T(c_t) = g(l_t). \quad (3)$$

Before building BP neural network for predicting components response time, we need to determine the topology structure and critical parameters of the network.

Layer. Hecht-Nielson proved that a hidden layer BP neural network could approximate any continuous function of a closed interval [11]. We use a BP neural network with one hidden layer.

Hidden Nodes. The number of hidden layer neurons affects the accuracy and speed of solving problem [12]. We use cut-and-trial method to determine the number of hidden layer neurons, and the empirical formula is

$$j = \sqrt{m+n} + a \quad a \in (1, 10), \quad (4)$$

in which m and n are the number of components, and a is a constant.

Activation Function. The hidden layer uses *tansig* sigmoid function, and the output layer uses *purelin* sigmoid function. *Tansig* is denoted as follows:

$$f(x) = \frac{1}{1 + e^{-x}}. \quad (5)$$

Its first-order backward function is denoted as follows:

$$f'(x) = f(x)(1 - f(x)). \quad (6)$$

Purelin function is denoted as follows:

$$g(x) = x. \quad (7)$$

Initial Weight and Threshold. Initial weight and threshold affect convergence ability and training time. In this paper, initial weight and threshold are random numbers between -1 and 1 .

Learning Rate. Learning rate decides the weight variation in the circuit training. We choose 0.01 as learning rate to ensure network stability.

Learning Algorithms. The standard BP algorithm has slow convergence speed problem and local minimum problem. In this paper, we use fast learning algorithm based on L-M (Levenberg-Marquardt) [13].

Error index function:

$$E(W) = \frac{1}{2} \sum_{k=1}^N e_i^2(W) = \frac{1}{2} \sum_{k=1}^N \sum_{i=1}^m (\hat{T}(c_i)_k - T(c_i)_k)^2; \quad (8)$$

$W(k)$ is the k time iterative network weight vector, and the weight adjustment formula is

$$W(k+1) = W(k) - [J^T J + \mu I]^{-1} J^T e. \quad (9)$$

Weighted differential Jacobian Matrix is J . Network error vector is e . Unit matrix is I . Coefficient is μ , which is adaptive adjustment.

If error index function $E(W)$ is reduced, we will use μ to calculate $W(k+1)$. Then we decrease μ ; otherwise we increase μ .

The set of the number of N components current requests and N components response time is $\{(x_k, y_k) \mid x \in R^m, y \in R^n, k = 1, 2, \dots, N\}$. The total samples are divided into training sample ϕ_1 and testing sample ϕ_2 :

$$\begin{aligned}\phi_1 &= \{(x_k, y_k) \mid x \in R^m, y \in R^n, k = 1, 2, \dots, N_1, N_1 \leq N\}, \\ \phi_2 &= \{(x_k, y_k) \mid x \in R^m, y \in R^n, k = N_1 + 1, N_1 + 2, \dots, N\}.\end{aligned}\quad (10)$$

First of all, we train N_1 groups component current requests and response time sample to establish mapping relation. Then, we use the number of component current requests from the $N_1 + 1$ th group to the N th group to predict component response time.

The training sample network error is denoted as follows:

$$e_k = \frac{1}{2} \sum_{i=1}^m (\hat{T}(c_i)_k - T(c_i)_k)^2. \quad (11)$$

The network total error is less than ε_1 :

$$E_1 = \frac{1}{2} \sum_{k=1}^{N_1} \sum_{i=1}^m (\hat{T}(c_i)_k - T(c_i)_k)^2 \leq \varepsilon_1. \quad (12)$$

Testing sample of mean square error is less than ε_2 :

$$E_2 = \frac{1}{N - N_1} \sum_{k=N_1+1}^N \sum_{i=1}^m (\hat{T}(c_i)_k - T(c_i)_k)^2 \leq \varepsilon_2. \quad (13)$$

Because we are not sure of the accurate number of concurrent requests and the sample response time. Even if E_1 is small, we still cannot ensure that E_2 is up to grade.

4.3. Genetic Algorithm Parameters Setup. Genetic algorithm optimizes BP network component response time prediction weight to overcome some shortages of BP neural network, so we can accurately predict component response time. Genetic algorithm for optimizing network weight needs to design the main parameters.

Encoding System. There are two chromosome encoding methods: binary encoding and real encoding. Real encoding is shorter than binary encoding, and it needs not switch back and forth between encoding and decoding [14, 15]. We use real encoding to improve computation efficiency.

Chromosome. Chromosome is the initial connection weight threshold vector of BP neural network. The numbers of hidden layer and input layer unit threshold are p, n . The BP network structure is $n-p-n$. The length of chromosome is $l = n \times p + p + p \times n + n$. The chromosome has l real numbers, and the scope of every real number is $(0, l)$. Mathematical expression of individual is

$$X = [w_{11}, w_{12}, \dots, w_{np}, \theta_1, \theta_2, \dots, \theta_p, v_{11}, v_{12}, \dots, v_{pn}, r_1, r_2, \dots, r_n]. \quad (14)$$

Number of Initial Population. If the number of initial population M is small, it will not benefit weight threshold crossing and variation and will decrease the diversity of population. But on the contrary, the running efficiency of genetic algorithm will decrease. Therefore, we decide M is 30.

Fitness Function. Fitness function is an index to decide the chromosome quality and is the only measure to guide the genetic algorithm search. Training overall error is as small as possible, and genetic algorithm uses the maximum objective function value as the fitness function. We calculate square of components response time error sum in the training sample according to formula (12), and chromosome adaptive value is

$$f_i = \frac{1}{E_i}; \quad (15)$$

$i = 1, 2, \dots, M$ is chromosome number.

Selection Operator and Selection Probability. We select the most fitness individual to continue genetic manipulation. In this paper, we adopt roulette wheel selection method. Fitness f_i selective probability is P_s :

$$P_s = \frac{f_i}{\sum_{j=1}^M f_j}. \quad (16)$$

Crossover Operator and Crossover Probability. We randomly select two chromosomes and choose their weights and thresholds with crossover probability to form two new individuals X_A^t, X_B^t . We use arithmetic crossover operation to generate two new individuals:

$$\begin{aligned}X_A^{t+1} &= \alpha X_B^t + (1 - \alpha) X_A^t, \\ X_B^{t+1} &= \alpha X_A^t + (1 - \alpha) X_B^t;\end{aligned}\quad (17)$$

$a \in (0, 1)$ is a parameter. We adopt nonuniform arithmetic crossover operator a . If the crossover probability value is large, the convergence speed will be fast. On the contrary, if the crossover probability value is small, algorithm performance will be reduced. In this paper, P_c is 0.4.

Mutation Operator and Mutation Probability. We randomly select a chromosome, and mutation probability P_m selects weight threshold to achieve mutation. Individual is $X = (x_1 x_2, \dots, x_k, \dots, x_t)$. Mutation point is x_k and the value range is $[U_{\min}^k, U_{\max}^k]$. After mutation operation, we achieve a new individual $X = (x_1 x_2, \dots, x'_k, \dots, x_t)$. The new genetic value x'_k is denoted as

$$\begin{aligned}x'_k &= x_k + (U_{\max}^k - x_k) \cdot f(g) \quad r > 0.5, \\ x'_k &= x_k - (x_k - U_{\min}^k) \cdot f(g) \quad r < 0.5.\end{aligned}\quad (18)$$

$f(g) = r_2 \cdot (1 - g/G \max)$. Random number is r_2 . The number of iterations is g . The max number of evolution is $G \max$. The value range of r is $[0, 1]$.

Input: The most similar virtual machine's number of component current requests and response time historical sequence is $N_i(c_1), N_i(c_2), \dots, N_i(c_n)$ $T_i(c_1), T_i(c_2), \dots, T_i(c_n)$ $i = 1, 2, \dots, t$ before time t . The number of component current requests is k .

Output: Component response time $T_i(c_1), T_i(c_2), \dots, T_i(c_n)$ $i = t + 1, t + 2, \dots, t + k$ in the future k time intervals.

- (1) Determine the number of concurrent requests, set the related parameters, establish the BP neural network.
- (2) Set up population size L , chromosome length M , number of iterations $Maxgen$, crossover probability P_c and mutation probability P_m . Initialize M sets of weighting value of the threshold. Form population.
- (3) Train the BP neural network. Calculate individual adaptiveness according to formula (15). Choose the optimal chromosome.
- (4) Use roulette wheel selection method to choose M the most adaptive individuals.
- (5) Choose several pairs of chromosomes in Step 4 and random number $r, r \in [0, 1]$ for every pair. If $r < P_c$, chromosome execute crossover operation.
- (6) Choose several pairs of chromosomes in Step 5 and random number $r, r \in [0, 1]$ for every pair. If $r < P_m$, chromosome execute mutation operation.
- (7) Choose the individual in Step 6 as network weight threshold to train BP network. Compare the optimal individual with the last generation optimal individual. If the former is more adaptive, it will replace the last generation optimal individual. Moreover, the last generation optimal chromosome replace the worst chromosome.
 $gen = gen + 1$, if $gen = Maxgen$, algorithm will be stoped. Then we get optimal weight threshold. Otherwise, go to Step 4.
- (8) Decompose the optimal individual to get BP neural network connective weight threshold.
- (9) BP network initial iteration value $i = 1$.
- (10) Put the amount of component concurrent requests and response time sequences into model, and calculate component response time according to formula (3).
- (11) Formula (12) calculates the first sample component response time prediction error quadratic sum and formula (9) adjusts weight.
- (12) Choose the second sample component current requests amount and response time. Repeat Steps 10 and 11 until finishing N samples training.
- (13) $i = i + 1$. if $i = N$ or the overall error of the network $E < \epsilon$. When training process is ended, we get future component response time. Otherwise, go to Step 10.
- (14) Put the number of k components current requests into trained network. Then get the new virtual machine component k response time.

ALGORITHM 1: Component response time prediction procedure.

If the mutation probability is too large, algorithm performance will be reduced. Instead, the species diversity will be poor. In this paper, P_m is 0.4.

Termination Condition. We adopt maximum evolution algebra as termination condition. In this paper, it is 100.

4.4. Service Response Time Calculation. According to the previous sections, Algorithm 1 shows the steps of component response time prediction procedure.

There are m components deployed on n virtual machines and q requests which are transmitted to component j on virtual machine k . Virtual machine response time $\bar{t}_{jk}$ is denoted as

$$\bar{t}_{jk} = \frac{1}{q} \sum_{i=1}^q t_i. \quad (19)$$

Calculate weighted sum of n virtual machines component response time. Therefore service response time is

$$t = \frac{\sum_{k=1}^n \sum_{j=1}^m \bar{t}_{jk}}{m \times n}. \quad (20)$$

5. Experimental Evaluation

5.1. Experimental Setup. We deploy six virtual machines on the server and two virtual CPUs on the virtual machine. Then, we deploy three procedures on the six virtual machine clients. Load balancer is deployed requests on virtual machine according to ratio and the client sends requests to the load balancer. Finally, we monitor the number of virtual machine current requests and the response time every 30 seconds and put the monitor data into database. After the server runs for several minutes, we add a new virtual machine.

5.2. GA-BP Neural Network Structure. When we calculate similarity, the weight vector was $\{0.5, 0.25, 0.15, 0.1\}$, which represented CPU, memory, I/O, and bandwidth. We found out that the most similarity virtual machine was VM3, and we used the number of VM3 component concurrent requests and response time to predict VM7 (the new VM) service component response time. We predicted VM7 component concurrent time by traditional BP algorithm and GA-BP algorithm and evaluated the new algorithm. There were 300 samples on VM3. 80% of the samples were training

TABLE 1: Net error of different hidden layer node number.

Hidden layer	5	6	7	8	9	10	11	12
Network general error	0.6906	0.7012	0.6893	0.3859	0.4984	0.2018	0.4991	0.5274

FIGURE 3: Performance curve of BP neural network training process.

samples and 20% of the samples were verification samples. Normalization formula is

$$\hat{x} = \frac{2(x - x_{\min})}{x_{\max} - x_{\min}} - 1. \quad (21)$$

Normalization data is $\hat{x}$, and current data is x .

5.2.1. BP Neural Network Structure. BP network training function is *trainlm*. Learning rate is 0.1. Performance function is MSE (mean square error). Training target error is 0.0001. The largest number of iterations is 1000. Adopt cut-and-try to determine optimal hidden layer node number. Table 1 shows experiment result. When hidden layer node number is 10, network has minimum error. The network structure is 5-10-1. Figure 3 shows BP neural network training performance when we train 240 samples. When BP neural network is trained 36 steps, the validation performance is 0.00012949.

5.2.2. GA-BP Neural Network Structure. We set the genetic algorithm parameters as follows. Population size is 30; evolution algebra is 100; crossover probability is 0.4; the mutation probability is 0.01; BP network structure is 5-10-1; performance function is MSE, and training target error is 0.0001. Figure 4 shows that when GA-BP neural network is trained 8 steps, the validation performance is 0.00012949. Convergence rate is faster than traditional BP neural network.

FIGURE 4: Performance curve of GA-BP neural network training process.

FIGURE 5: Three traditional methods and GA-BP neural network prediction error.

5.3. Result Analysis of Comparative Experiment. We use GA-BP neural network model to predict 21 components on 7 virtual machines. The next service response time prediction value is 0.985 s, and the real response time is 0.997 s. Under the same experiment condition, we also use BP neural network, time series, and queuing theory to predict service response time in 60 seconds. Figure 5 shows every method prediction error. When prediction time is less than real time, the error is a negative and vice versa. Figure 6 compares GA-BP neural network prediction response time with real response time. Figure 7 shows the differences between BP neural network prediction response time and real response

FIGURE 6: Comparison between GA-BP neural network prediction value and real value.

FIGURE 8: Comparison between time series prediction value and real value.

FIGURE 7: Comparison between BP neural network prediction value and real value.

FIGURE 9: Comparison between queuing theory prediction value and real value.

time. Figures 8 and 9 show traditional time series and queuing theory prediction response time in contrast to real response time. Time series and queuing theory are not as accurate as GA-BP neural network and BP neural network, because BP neural network could reduce the error between the predicted values and real value and increasing speed of system processing. What is more, GA-BP neural network overcomes local optimal solution and approximate optimal in a short period. Therefore, it is more stable than other methods in the whole prediction process.

6. Conclusion and Future Work

Cloud environment dynamic scalability will break the law of service response time. Traditional method cannot satisfy prediction accuracy requirements. In this paper, we propose a novel method to predict virtual machine response time based on GA-BP neural network. We predict components response time on the VM and calculate virtual machine service response time. The extensive experimental results show the effectiveness and efficiency of our framework.

For future work, we will investigate more techniques for improving the prediction accuracy.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work is supported by grants from National Natural Science Foundation of China (no. 61300019).

References

- [1] M. Armbrust, A. Fox, R. Griffith et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.
- [2] M. Alhamad, T. Dillon, and E. Chang, "Conceptual SLA framework for cloud computing," in *Proceedings of the 4th IEEE International Conference on Digital Ecosystems and Technologies (DEST '10)*, pp. 606–610, Dubai, United Arab Emirates, April 2010.
- [3] G. Box, G. Jenkins, and G. Reinsel, *Time Series Analysis*, Holden-day, San Francisco, Calif, USA, 1970.

- [4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [5] X. Zheng, J. Zhao, and Z. Cheng, "Web service response time dynamic prediction approach," *Journal of Chinese Computer System*, no. 8, pp. 1570–1574, 2011.
- [6] Y.-J. Chiang and Y.-C. Ouyang, "Profit optimization in SLA-aware cloud service with a finite capacity queuing model," *Mathematical Problems in Engineering*, vol. 2014, Article ID 534510, 11 pages, 2014.
- [7] N. U. Bhar, *An Introduction to Queuing Theory Model and Analysis in Application*, Birkhauser, Boston, Mass, USA, 2007.
- [8] K. Xiong and H. Perros, "Service performance and analysis in cloud computing," in *Proceedings of the IEEE World Congress on Services*, pp. 693–700, Los Angeles, Calif, USA, July 2009.
- [9] J. Yi, Q. Wang, D. Zhao, and J. T. Wen, "BP neural network prediction-based variable-period sampling approach for networked control systems," *Applied Mathematics and Computation*, vol. 185, no. 2, pp. 976–988, 2007.
- [10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.
- [11] H. R. Nielson, *Neurocomputing*, Addison Wesley Publishing Company, Boston, Mass, USA, 1991.
- [12] J. S. Judd, *Neural Network Design & the Complexity of Learning*, California Instruction of Technology, Cambrige, UK, 1988.
- [13] N. Yamashita and M. Fukushima, "On the rate of convergence of the levenberg-marquardt method," *Computing*, vol. 15, pp. 239–249, 2001.
- [14] C. Z. Janikow and Z. Michalewicz, "An experimental comparisons of binary and floating point representations in genetic algorithms," in *Proceeding of the 9th Conference on Genertic Algorithm*, pp. 31–36, 1991.
- [15] Z. Michalewicz, C. Z. Janikow, and J. B. Krawczyk, "A modified genetic algorithm for optimal control problems," *Computers and Mathematics with Applications*, vol. 23, no. 12, pp. 83–94, 1992.