Future Generation Computer Systems 81 (2018) 41–52

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Workload prediction in cloud using artificial neural network and adaptive differential evolution

Jitendra Kumar [*], Ashutosh Kumar Singh

Department of Computer Applications, National Institute of Technology, Kurukshetra, India

HIGHLIGHTS

- The paper presents a workload prediction approach for cloud datacenters using neural network and self adaptive differential evolution.
- The proposed approach outperforms well known back propagation network approach in accuracy.
- The root mean squared error is used as accuracy measurement metric and proposed approach is able to achieve significant reduction in the prediction error.

ARTICLE INFO

Article history:
Received 10 January 2017
Received in revised form 26 July 2017
Accepted 22 October 2017
Available online 5 December 2017

Keywords:
Cloud computing
Data center
Workload prediction
Neural network
Differential evolution

ABSTRACT

Cloud computing has drastically transformed the means of computing in recent years. In spite of numerous benefits, it suffers from some challenges too. Major challenges of cloud computing include dynamic resource scaling and power consumption. These factors lead a cloud system to become inefficient and costly. The workload prediction is one of the variables by which the efficiency and operational cost of a cloud can be improved. Accuracy is the key component in workload prediction and the existing approaches lag in producing 100% accurate results. The researchers are also putting their consistent efforts for its improvement. In this paper, we present a workload prediction model using neural network and self adaptive differential evolution algorithm. The model is capable of learning the best suitable mutation strategy along with optimal crossover rate. The experiments were performed on the benchmark data sets of NASA and Saskatchewan servers' HTTP traces for different prediction intervals. We compared the results with prediction model based on well known back propagation learning algorithm and received significant improvement. The proposed model attained a shift up to 168 times in the error reduction and prediction error is reduced up to 0.001.

1. Introduction

Cloud computing also referred as on-demand computing is expanded remarkably in the past few years. Companies are getting their applications up and running faster through the cloud. It has enabled users to store and process their data in third party data centers. Usage of virtual computing resources that are available as services over the Internet has gained significant attention from both industry and academia. In traditional computing, a user can have access to a fixed amount of computing resources. On the other hand, in cloud computing approach on demand resources are being offered to its customers [1]. Thus cloud has been very helpful in avoiding upfront infrastructure costs of companies. A wide number of organizations are switching to the cloud because of its characteristics like robustness, scalability, on demand service and several others. Service providers deploy huge data centers in order to provide on demand services to their customers. These data centers work as the backbone of the cloud computing environment. The key concept behind the cloud data centers is virtualization that helps in sharing resources among multiple users through virtual machines (VM). In order to maximize the gain of service providers along with maintaining the quality of services (QoS), data centers need an efficient and dynamic resource scaling and allocation policy.

The possibility of dynamic scaling along with advanced resource management is raised due to the flexibility of adding or removing virtual hardware resources at any point of time in the lifetime of hosted application [2]. Dynamic resource scaling has become a critical point of concern for a data center to work in a flawless manner. Resource scaling depends on several factors including number of active users, upcoming events, the current

* Corresponding author.
E-mail addresses: jitendrakumar@ieee.org (J. Kumar), ashutosh@nitkkr.ac.in (A.K. Singh).

https://doi.org/10.1016/j.future.2017.10.047
0167-739X/© 2017 Elsevier B.V. All rights reserved.

state of the system and several others. The reactive scaling methods are not beneficial due to non instantaneous initialization and migration of virtual machines. While the proactive methods allow scaling of resources before actual demand arrives but these approaches require upcoming workload information in advance. The future resource demands can be predicted through identification of historical usage patterns and the current state of the data center. Since the predicted load on data center will determine the amount of resources to scale, it is required to set up an effective and reliable prediction system to achieve an accurate estimation of upcoming workload. This information can also be used to improve resource utilization and power consumption. Thus, the cloud data center can be operated at lower cost along with reducing SLA violations instances. The key challenges for precise predictions are interaction with varying number of clients and high non-linearity in workload. The workload can be predicted through several approaches. One can measure the maximum or average workload for specified time intervals. However, methods based on the statistics such as mean and maximum are very general and not capable of producing accurate predictions. For instance, if we use a prediction model based on maximum workload then resources will remain unused most of the time. On the other hand, if the average approach is used to predict future workload then the system will witness the lack of resources. This will cause performance degradation whenever workload increases. The prediction models based on such methods are considered to be poor as they are not good in prediction and correspond to a very few number of cases [3]. Machine learning methods are widely being adopted for establishing more accurate prediction models. Machine learning techniques use historical data as training window to predict the workload throughout a prediction interval [4]. Where prediction interval defines the time between each prediction.

This paper contributes towards the development of workload prediction model based on neural network and self adaptive differential evolution that can predict the workloads with higher accuracy. Instead of using simple statistics such as mean and others, the proposed model learns and extract the pattern from workload. The obtained patterns are used for further predictions. The model is trained using evolutionary approach to minimize the effect of initial solution choice. The evolutionary algorithm explores the space in multiple directions using a set of solutions. One of the difficult tasks in evolutionary algorithms is the parameter tuning. This effect is minimized as the proposed predictive model is capable in learning the crossover rate, mutation rate and mutation strategy too. These predictions can be further utilized in improving the resource scaling decisions. The model has been tested on the benchmark data sets of two servers and compared with the back propagation neural network model. The proposed model outperformed the prediction models based on average, maximum and back propagation network with a substantial reduction in mean squared prediction error.

Rest of the paper is organized as follows: Section 2 provides an overview of related work. The proposed approach is discussed in Section 3 followed by results and discussion in Section 4. Finally, the paper is wrapped up with conclusive remarks and future scope in Section 5.

2. Related work

Many of the researchers are working in the area of workload prediction and they have addressed the problem with the help of different approaches. Two well known approaches are homeostatic prediction and history-based prediction. In homeostatic prediction, the upcoming workload at next time instance is predicted by adding or subtracting a value such as the mean of previous workloads from the current workload [5]. The value to be subtracted or added can be static (a fixed number) or dynamic (estimated value from previous workload instances). On the other hand, history based methods are simple and popular prediction models. These models analyze previous workload instances and extract patterns to predict the future demands. History based methods use tendency information of previous workload instances while homeostatic model goes back to the mean of previous workloads [6].

In order to use historical information, data center workloads exhibiting seasonal trends can be presented in the form of time series [7]. A set of data points measured at successive points in time spaced at uniform time intervals is called time series data. Classical methods have been used extensively in time series prediction. This group of models includes Exponential Smoothing (ES), auto regression (AR) model, Moving Average (MA) model, Autoregressive Integrated Moving Average (ARIMA) model, Hidden Markov Model (HMM) and many others.

In [8], the authors have proposed an auto regression based method to predict the web server workload but the model is strictly linear in nature. In auto regression, a linear combination of past values of the variable under consideration is used to forecast the value for upcoming time instances. Danilo et al. [9] used moving average to develop a prediction model. A distributed solution was proposed that incorporated workload prediction and distributed non linear optimization techniques. In [10], Kalekar used exponential smoothing for seasonal time series prediction. Two different approaches multiplicative seasonal model and additive seasonal model were used to predict the workload. The model is appropriate for time series exhibiting seasonal behavior only. Roy et al. [3] proposed a forecasting model using Auto Regressive Integrated Moving Average (ARIMA) model. The authors also discussed the challenges involved in auto scaling in a cloud environment. In [11], the authors have used regression techniques to analyze live sports event broadcast service workloads from a commercial Internet service provider. The approach is based on simple statistical models that might not capture the patterns in more complex data. Khan et al. [12] presented a workload prediction model based on multiple time series approach. The model does a grouping of similar applications' need in order to improve the accuracy of predictions. The authors also utilized hidden Markov model (HMM) to distinguish the temporal correlations in obtained clusters of VMs. They use this information to define the variations in workload patterns over time. Other methods also have been used in workload forecasting such as Monte Carlo [13].

Apart from classical approaches machine learning also has been widely explored for time series prediction. Machine learning methods can learn from data provided to them and give some probabilistic score on an unknown pattern from its past experience. In [14] authors proposed a framework using self organizing map (SOM) and support vector machines (SVMs). Self organizing map was used to cluster the data in different regions while SVMs were used to predict the future data. But the approach is highly sensitive to the threshold value that decides the number of data points in a partitioned region. A two tier architecture using k Nearest Neighbors (kNN) was proposed by Tao Ban et al. for financial time series prediction [15] but $kNNs$ are lazy learners and need high computational cost. The Neural network was used in [16] to model workload variations in multimedia designs. In [17] authors presented a resource scaling method based on workload prediction. Linear regression was used for predicting workload. The predicted workload was used to decide the type of scaling.

Hu et al. [18] used the statistical learning theory to build a prediction model that integrated with support vector regression (SVR) and Kalman smoother. It achieved high prediction accuracy compared to the auto regression, back propagation network and standard SVR. The predictions were used for further resource scaling decisions. In [19] authors implemented an ensemble based workload prediction model. It uses five different base prediction models. Each model is assigned a weight and contributes accordingly in predictions. The weights are assigned and optimized using genetic algorithm. The proposed model suffers from slower execution time. Islam et al. [20] presented a workload prediction model based on neural network and linear regression. They also proposed the concept of the sliding window. The model was evaluated using CPU demand traces and found better in comparison to non sliding window based neural network and linear regression models. In [21], Tran et al. used seasonal ARIMA model to propose a workload prediction model capable of predicting workload ahead up to 168 h. The proposed model seeks for an expert guidance in order to optimize the order of model. Prevost et al. [22] worked on a prediction model based on the back propagation neural network. It utilizes the stochastic model to strengthen the accuracy of predictions. Caron et al. [23] used pattern matching concept to predict upcoming workload. The model used KMP string matching algorithm to find similar patterns in the past and predicts based on the patterns from history. The model becomes slower as the history of workloads increases. Fuzzy Neural Network was used in [24], authors presented each resource as a period of flatness or fluctuation. The system is complex as two layers of predictions are involved that introduce the time delay in workload prediction. Gong et al. presented a resource scaling policy named PRESS that monitored virtual machines' CPU usage [25]. Chang et al. [26] proposed a workload prediction model using neural network and steepest descent learning algorithm. The model improved prediction accuracy over time delay neural network and linear regression but the proposed method also suffers from high prediction errors. The models based on gradient based approaches are sensitive towards the choice of learning rate and initial solution. To overcome the problem of fixed learning rate, Lu et al. [27] presented a back propagation learning algorithm based workload prediction model for cloud environment. It adjusts the weights of model according to error trend. They evaluated the model over Google cluster trace by forecasting computational latency based tasks. The accuracy of model drops down as the latency levels are increased. In the proposed approach we are using neural network with self adaptive differential evolution as a tool to process the historical data and predict the upcoming load on the data center. Evolutionary approaches avoid the effect of choice of initial solution by exploring the solution space in multiple directions through a set of solutions distributed uniformly in solution space. They also suffer with the choice of parameter values such as mutation and crossover rates. The proposed approach also provide a solution to this by learning the parameter values and mutation strategy as well. Based on the predicted load improved resource scaling decisions can be taken in advance to avoid SLA infringements.

3. Workload prediction approach

This section focuses on the working of the proposed prediction model. The output of workload predictor along with the current state of the data center is passed to a device called resource manager as shown in Fig. 1. This information can be used effectively by the resource manager for further scaling decisions.

3.1. Prediction model

The prediction approach involves a number of steps as shown in Fig. 2. In preprocessing step, first we extract the requests and aggregate them into a time unit level (we consider minute level in this paper). Following aggregation, data X is normalized in the range $(0, 1)$ through Eq. (1). Where X_{min} and X_{max} are minimum and maximum values respectively obtained from data set. The value of normalized data $\hat{X}$ is fed into the network as input. This is followed by training and evaluation of the network along with workload prediction.

$$\hat{X}_i = \frac{X_i - X_{min}}{X_{max} - X_{min}}. \tag{1}$$

The predictive model is composed of three layered neural network as shown in Fig. 3. The neural network is a combination of logical units called neurons [28]. In our approach, we have used an architecture with $n - p - q$ neurons. Where n, p and q are numbers of neurons in input, hidden and output layer respectively. The value of q is chosen to be 1 while a number of experiments were performed with different combinations of n to decide the architecture of the network. The opted values for various parameters in experiments are mentioned in Table 5. The linear and sigmoid activation functions are used for different nodes. The activation function used for a node can be identified from Eq. (2). The supervised learning approach is opted to train the model. The detailed discussion on training is done in Section 3.2. The predictive model extracts patterns from actual workload and analyzes n previous workload values in order to predict upcoming workload on the data center at time instance $n + 1$.

$$f(x) = \begin{cases} x & \text{if input layer} \\ \dfrac{1}{1+e^{-x}} & \text{otherwise.} \end{cases} \tag{2}$$

In order to utilize neural network model for workload prediction, the number of requests measured over time are stored as historical data and used as input. The input data (X) and corresponding output (y) are constructed as shown in Eq. (3). Where each X_i (ith row of X) denotes one data point with n features and x_j represents the number of actual requests received on jth time instance. For example, let R denotes a set of workloads measured over time t and $\hat{R}$ is normalized form of R. If only 5 previous values are considered for predicting the upcoming workload at next time instance, i.e. $n = 5$, the input vector can be formed as shown in Eq. (4).

$$X = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ x_2 & x_3 & \cdots & x_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_k & x_{k+1} & \cdots & x_{n+k-1} \end{bmatrix}, y = \begin{bmatrix} x_{n+1} \\ x_{n+2} \\ \vdots \\ x_{n+k} \end{bmatrix} \tag{3}$$

$$R = \{65, 3, 84, 93, 67, 75, 74, 39, 65, 17, 70, 3, 27, 4, 9, 82, 69, 31, 95, 3\}$$
$$\hat{R} = \{0.67, 0, 0.88, 0.98, 0.70, 0.78, 0.77, 0.39, 0.67, 0.15, 0.73, 0, 0.26, 0.01, 0.07, 0.86, 0.72, 0.30, 1.00, 0\}$$

$$X = \begin{bmatrix} 0.67 & 0 & 0.88 & 0.98 & 0.70 \\ 0 & 0.88 & 0.98 & 0.70 & 0.78 \\ 0.88 & 0.98 & 0.70 & 0.78 & 0.77 \\ 0.98 & 0.70 & 0.78 & 0.77 & 0.39 \\ 0.70 & 0.78 & 0.77 & 0.39 & 0.67 \\ 0.78 & 0.77 & 0.39 & 0.67 & 0.15 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}, y = \begin{bmatrix} 0.78 \\ 0.77 \\ 0.39 \\ 0.67 \\ 0.15 \\ 0.73 \\ \vdots \end{bmatrix}. \tag{4}$$

The input data is further divided into two different parts called training and testing data. Where training data is used to train the prediction model while testing data is applied on model for evaluating prediction accuracy. We used 60% of data as training data while 40% data is used for the purpose of testing. Following the training of model, its accuracy over unseen patterns is evaluated and measured using square root of mean squared error (MSE) Eq. (5). Where y_i and $\hat{y}_i$ are the actual and predicted workloads

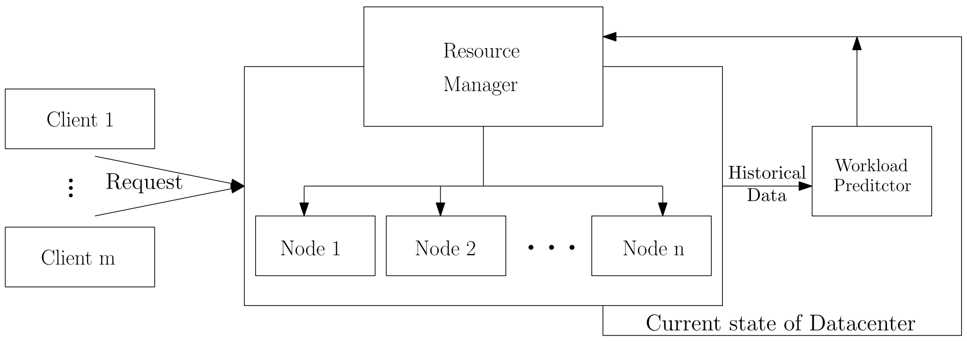


Fig. 1. Cloud data center architecture.

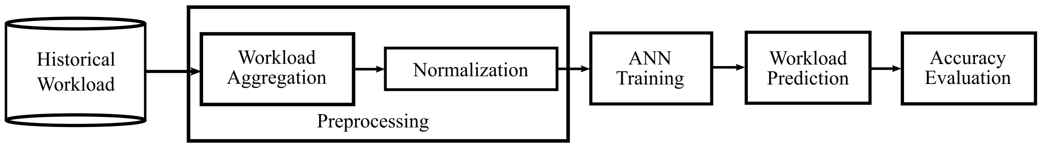


Fig. 2. Workload prediction approach work flow.

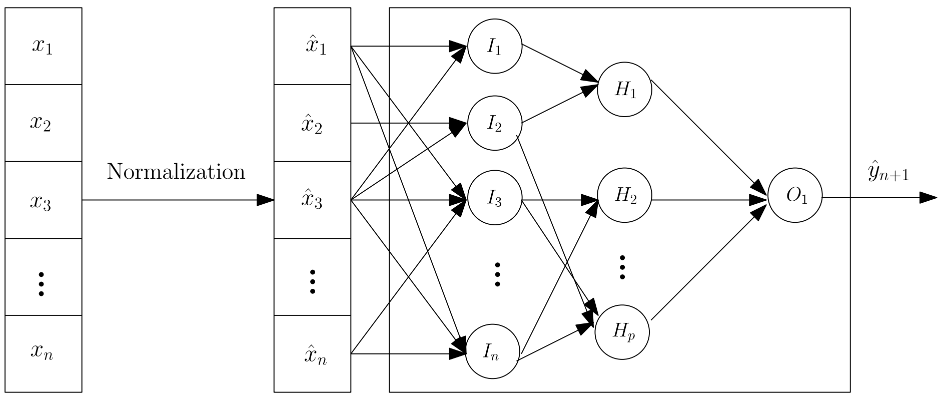


Fig. 3. Workload predictor model.

respectively at time instance i, while m is the total number of samples under consideration. Operational summary of the workload predictor is shown in Algorithm 1 .

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2. \tag{5}$$

3.2. *Training of neurons*

Neural networks provide a distinct form for data analysis, pattern extraction and recognition than traditional computing methods. Even they are considered as an expert in their domain, they require training. The Differential Evolution (DE) developed by Price and Storn in 1995 is a reliable technique to train neural network [29]. Since its first publication in 1995, DE has proved itself in various competitions among evolutionary algorithms like IEEE evolutionary computation. The model is trained using self adaptive differential evolution (SaDE). Self adaptive differential evolution is one of the differential evolution's variants [30]. It has been widely used to train feed forward neural network for different applications such as [31]. We chose this algorithm because it is simple, easy to use and it has outperformed other population based approaches such as PSO, GA and others [32–36]. Unlike differential evolution, SaDE is capable of learning optimal crossover rate and best suitable mutation strategy.

3.2.1. *Initialization*

Similar to basic differential evolution, SaDE also begins with N data points generated randomly with uniform distribution as shown in Eq. (6). Here each data point is a real valued vector of size $((n+1)\times p+p\times 1) \Rightarrow p(n+2)$ representing synaptic weights of the prediction model. The number of neurons in input layer become $n+1$ after considering one additional bias neuron.

$$X_{i,j} = lb_j + rand \times (ub_j - lb_j) \tag{6}$$

where $lb_j = -1$ and $ub_j = 1$ are the lower and upper bounds respectively. *rand* is random number in the range [0, 1].

3.2.2. *Mutation*

The algorithm is capable of learning the optimal mutation rate (F) and the best suitable strategy. Initially a vector F_v of length N is initialized in the range (0, 1] with normal distribution of mean ($F_\mu = 0.5$) and standard deviation ($F_\sigma = 0.3$) [30]. Where low and higher values deal with local and global search respectively and F_i represents the mutation rate for ith vector in population.

We selected three mutation policies for our experiments, namely *DE/rand/1*, *DE/best/1* and *DE/current to best/1*. Each of them is illustrated in Eqs. (7) to (9) respectively. However, this can be extended to any number of mutation strategies. *DE/Rand/1* was picked because it is good in maintaining diversity while *DE/current*

$$p_1 = \frac{st_1(st_2 + ft_2 + st_3 + ft_3)}{2(st_2st_3 + st_1st_3 + st_1st_2) + ft_1(st_2 + st_3) + ft_2(st_1 + st_3) + ft_3(st_1 + st_2)} \tag{11}$$

$$p_2 = \frac{st_2(st_1 + ft_1 + st_3 + ft_3)}{2(st_2st_3 + st_1st_3 + st_1st_2) + ft_1(st_2 + st_3) + ft_2(st_1 + st_3) + ft_3(st_1 + st_2)} \tag{12}$$

$$p_3 = 1 - (p_1 + p_2). \tag{13}$$

Box I.

to best/1* emphasizes on convergence property [30]. Montes et al. showed that *DE/best/1* is good for optimization problems [37].

$$DE/rand/1 \qquad u_i^k = x_{r_3}^k + F_i \times (x_{r_1}^k - x_{r_2}^k) \tag{7}$$

$$DE/current\ to\ best/1$$
$$u_i^k = x_i^k + F_i \times (x_{best}^k - x_i^k) + F_i \times (x_{r_1}^k - x_{r_2}^k) \tag{8}$$

$$DE/best/1 \qquad u_i^k = x_{best}^k + F_i \times (x_{r_1}^k - x_{r_2}^k). \tag{9}$$

In Eqs. (7) to (9), u_i^k and x_i^k represent ith trial solution generated by mutation operation and solution in current population of kth iteration respectively. r_1, r_2 and r_3 are mutually distinct random numbers in the range of $[1, N]$. x_{best}^k is the best solution found so far up to kth iteration. And F_i represents the mutation rate for corresponding vector.

Let us assume that p_1, p_2 and p_3 are the probabilities of applying *DE/rand/1*, *DE/current to best/1* and *DE/best/1* mutation strategies respectively. In our experiments, initially we defined $p_1 = p_2 = 0.33$ and $p_3 = 0.34$ to provide more or less equal chances to each mutation strategy to be applied. Roulette wheel selection is implemented to select the mutation strategy based on given probabilities. For this a vector *msp* of N random numbers is generated in the range $(0, 1)$ that defines the probabilities of mutation strategy selection for each vector. We select mutation policy based on the selection probability, e.g. if $msp(i)$ is less than or equal to p_1, *DE/rand/1* is applied. If the number is larger than p_1 and less than equal to $(p_1 + p_2)$, *DE/current to best/1* strategy is applied. And if number does not fall into either of the above cases, we apply *DE/best/1*. The mutation selection procedure is summarized in Eq. (10).

$$MS_i = \begin{cases} DE/rand/1 & 0 < msp(i) \le p_1 \\ DE/\text{current to best}/1 & p_1 < msp(i) \le (p_1 + p_2) \\ DE/best/1 & \text{otherwise.} \end{cases} \tag{10}$$

In order to find the best suitable mutation strategy, the information of each offspring solution generated after crossover operation whether successfully reaching in next iteration or not is stored. The number of offspring solutions successfully made into next generation generated by each mutation strategy are stored in st_1, st_2 and st_3 respectively. Similarly the number of offspring networks failed in reaching next iteration are stored in ft_1, ft_2 and ft_3. The percentage of success rate of offspring vectors generated by mutation strategies *DE/rand/1*, *DE/current to best/1* and *DE/best/1* is represented by Eqs. (11) to (13) respectively given in Box I.

After a fixed number of iterations called mutation strategy learning period (lp^F) the probabilities get updated (in our case it was 10). The effect of different values of learning period ranging from 1 to 12 is depicted in Fig. 5. The values of st_1, st_2, st_3, ft_1, ft_2, and ft_3 get reset after update in probabilities. This method allows to learn an appropriate mutation policy along with the mutation rate.

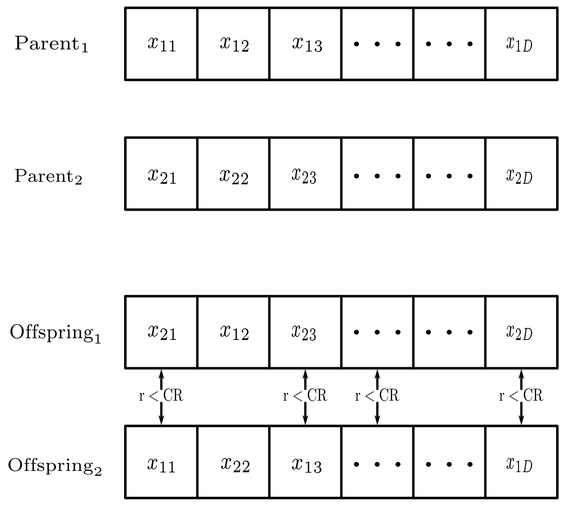

Parent$_1$
x_{11} x_{12} x_{13} ... ... x_{1D}
Parent$_2$
x_{21} x_{22} x_{23} ... ... x_{2D}
Offspring$_1$
x_{21} x_{12} x_{23} ... ... x_{2D}
r < CR
r < CR
r < CR
r < CR
Offspring$_2$
x_{11} x_{22} x_{13} ... ... x_{1D}

Fig. 4. Uniform crossover.

3.2.3. Crossover

Following mutation operation, the crossover is applied to generate new solutions called offsprings or children v_i where $i = (1, \ldots, N)$. Similar to the mutation rate, crossover rate (CR_v) was initialized randomly for each solution of population in the range of $(0, 1]$ with normal distribution of mean (CR_μ) 0.5 and standard deviation (CR_σ) 0.1 [30]. Where CR_i denotes the crossover rate for ith solution in the population.

In our approach uniform crossover (see Fig. 4, where D denotes the dimension of solution) is chosen because it allows each parent to contribute at gene level rather than segment level. In this approach, a random number in the range of (0, 1) is generated for each gene. If the random number generated for jth gene of ith solution is less than crossover rate CR_i, the values of $x_{i,j}$ and $u_{i,j}$ get exchanged. One randomly selected gene (j_{rand} in the range of $[1, D]$) is forced to swap the values to make sure the newly generated solutions are dissimilar than their parents. The crossover rate also gets reinitialized after a fixed number of generations (lp^{CR}). The value of CR associated with the offspring solutions successfully entering into next iteration was recorded and after a fixed number of iterations (g_{lp}^{CR}) mean is recalculated using recorded values of CR. The recorded values are reset after recalculation of the mean. Thus the crossover rate is optimized.

3.2.4. Selection

After generating offspring solutions, the population for next generation or iteration is selected. And this selection is done using survival of the fittest between x_i and v_i as shown in Eq. (14) .

$$x_i^{k+1} = \begin{cases} v_i^k & \text{if} f(v_i^k) < f(x_i^k) \\ x_i^k & \text{otherwise.} \end{cases} \tag{14}$$

3.2.5. An example

Let us consider a network with 3 input, 2 hidden and 1 output nodes to illustrate one iteration of training process. Total number of interconnections or length of vector can be calculated as $((3+1)\times 2+2\times 1)=10$. Let we have 4 vectors, each representing one network in our population X and initial values $(X^{(1)})$ are as given in Table 1.

Since we have 4 members in our population, the length of msp, F_v and CR_v vectors will be 4. The initial values are:

	msp	F_v	CR_v
1	0.31000	0.00001	0.55000
2	0.62000	0.27000	0.65000
3	0.84000	0.08000	0.68000
4	0.35000	0.38000	0.49000

Once initialization is done, each member of the population is evaluated through objective function. Since each member represents a network used to predict workload, the evaluation function is the root of mean squared error. This metric is used to evaluate the performance of the network or the prediction accuracy. Let $f(x_i)\in[0,1]$ represents the obtained root mean squared error for ith population member. The objective function is the minimization of root mean squared error. The fitness values for each member of initial population is given below:

Initial Population's Fitness $(f(X^{(1)}))$			
0.24	0.35	0.49	0.27

Followed by fitness evaluation, the mutation is performed to generate mutant vectors U_i where $i=(1\ldots N)$. In this example, first and last mutant vectors are generated using DE/rand/1 while second and third members are mutated using DE/current to best/1 and DE/best/1 respectively. The obtained mutated vectors are mentioned in Table 2.

Mutation operation is followed by crossover operator illustrated in Fig. 4. The offspring vectors obtained by crossover operation are given in Table 3 and their fitness values are mentioned below:

Offspring Population's Fitness $f(V^{(1)})$			
0.47	0.24	0.40	0.28

Next step after offspring generation and their evaluation is the selection as mentioned in Eq. (14). In this example, the members for next iterations are selected as $X^{(2)}=(x_1^{(1)},v_2^{(1)},v_3^{(1)},x_4^{(1)})$. The complete values for $X^{(2)}$ are shown in Table 4.

3.3. Time complexity

This section, presents the time complexity of SaDE based workload prediction approach. The first step (line 2) initializes the population of N networks each of length D and it requires $\mathcal{O}(N\times D)$. By putting $D=p(n+2)$, it becomes $\mathcal{O}(N\times p\times(n+2))\Rightarrow\mathcal{O}(n\times p\times N)$. Since $p<n$, it can be represented as $\mathcal{O}(n^2\times N)$. Next, line 3 initializes two vectors (CR_v and F_v) each of length N and needs $\mathcal{O}(2\times N)\Rightarrow\mathcal{O}(N)$. The next step (line 4) is fitness evaluation that calculates the average of squared differences between actual and predicted values of m samples over each network. It takes $\approx\mathcal{O}(n^2)$ in calculating predicted value for one data point over one network. The total amount of time taken to evaluate m data points over N networks is $\mathcal{O}(m\times n^2\times N)$. Line 7 initializes a vector of length N and takes $\mathcal{O}(N)$. Line 9 consumes $\mathcal{O}(N)$ in generating N times four random numbers. Line 10 to 18 are responsible for generating mutant and offspring solutions. It consumes $\approx\mathcal{O}(n^2)$ to generate one offspring solution using mutation and crossover operations. It will take $\mathcal{O}(n^2\times N)$ to produce N offspring solutions. Line 20 requires same set of instructions as line 4, that needs $\mathcal{O}(m\times n^2\times N)$.

Algorithm 1 Operational Summary of Workload Prediction Model

```
1: Initialize $CR_\mu = F_\mu = 0.5$, $CR_\sigma = 0.1$, $F_\sigma = 0.3$, $p_1 = p_2 = 0.33$, $p_3 = 0.34$
2: Randomly initialize $N$ networks of length $D$ (population)   ▷ Here $D$ is number of connections in network
3: Generate crossover $CR_v$ and mutation $F_v$ vectors (length $N$) $CR_v$, $F_v \in (0,1]$
4: Evaluate each network on training data using objective function
5: do
6:     for each generation $g$ do
7:         Generate a vector $msp$ of $N$ random numbers $\in (0,1]$
8:         for each solution $i$ do
9:             Generate $r_1 \neq r_2 \neq r_3 \neq i \in [1,N]$ and $j_{rand} \in [1,D]$
10:            if $0 < msp_i \le p_1$ then
11:                Apply DE/rand/1 mutation strategy and crossover
12:            else
13:                if $p_1 < msp_i \le (p_1 + p_2)$ then
14:                    Apply DE/current to best/1 mutation strategy and crossover
15:                else
16:                    Apply DE/best/1 mutation strategy and crossover
17:                end if
18:            end if
19:        end for
20:        Evaluate offspring vectors using objective function
21:        Select participants for next generation from offspring vectors and population
22:        Update values of $st_1, st_2, st_3, ft_1, ft_2$ and $ft_3$
23:        Update $p_1, p_2$ and $p_3$          ▷ (After each $lp^F$ generations)
24:        Regenerate $CR_v$                    ▷ (After each $lp^{CR}$ generations)
25:        Recalculate $CR_\mu$                 ▷ (After each $g_{lp}^{CR}$ generations)
26:    end for
27: while termination criteria is not met
```

The next step i.e. selection (line 21) requires N comparisons that need $\mathcal{O}(N)$. Line 22 and 23 elapsed constant time $\mathcal{O}(1)$. Line 24 and 25 takes $\approx\mathcal{O}(N)$ to regenerate CR_v and CR_μ respectively.

After, summing up all computed complexities it becomes $\mathcal{O}(m\times n^2\times N)$ per generation. The total time complexity for maximum number of generations (g) will become $\mathcal{O}(g\times m\times n^2\times N)$. We compared the complexity of SaDE based workload prediction approach with DE, GA and BPNN. The complexity of both DE and GA was founded in similar order as $\mathcal{O}(g\times N)$ is the only additional time required in the proposed approach. We found that the time complexity of BPNN based workload prediction approach is lesser than the proposed approach by a factor of N. The time complexity of BPNN based approach is $\mathcal{O}(g\times m\times n^2)$. But the proposed approach achieved an improvement in the accuracy up to 168 times over BPNN as discussed in Section 4 (Table 6).

4. Results and discussion

The experiments were performed on a machine equipped with Intel® Core$^{\text{TM}}$ I7-4790 processor of 3.60 GHz clock speed along with 4 GB of memory. We opted MATLAB as a tool for implementation and simulations. The data used for experiments are NASA HTTP traces and Saskatchewan HTTP traces, obtained from [38]. We call these data sets D_1 and D_2 respectively. NASA HTTP contains two traces having two month's worth of all HTTP requests to the NASA Kennedy Space Center WWW server in Florida. Saskatchewan HTTP data is seven months of HTTP logs from a University WWW server. The traces are stored in ASCII files with one line per request.

Table 1
Initial Population ($X^{(1)}$).

0.49	−0.58	−0.67	0.41	0.36	−0.76	−0.20	−0.96	−0.67	−0.12
−0.79	−0.80	0.33	−0.69	−0.93	0.05	−0.17	0.85	0.84	−0.48
0.36	0.65	0.79	0.91	0.62	−0.35	−0.64	0.31	0.59	0.50
−0.07	−0.65	0.03	0.08	0.50	0.09	−0.49	0.87	0.15	−0.54

Table 2
Mutant Vectors ($U^{(1)}$).

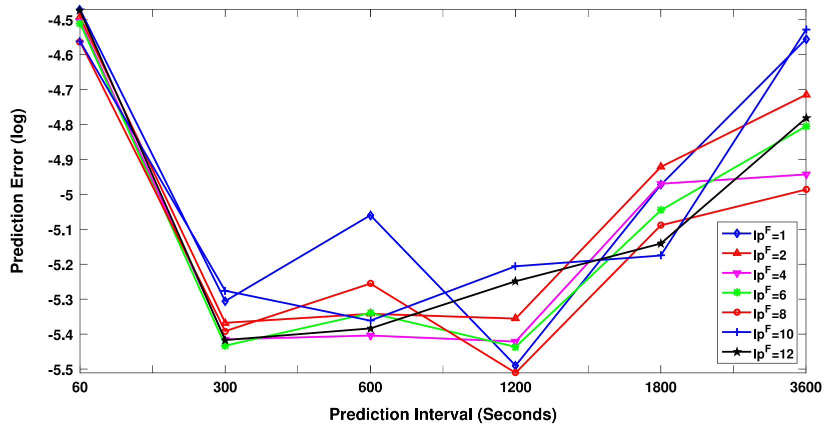
0.36	0.65	0.79	0.91	0.62	−0.35	−0.64	0.31	0.59	0.50
0.60	−0.23	−0.47	0.63	0.39	−0.88	−0.24	−1.00	−0.56	0.16
0.44	−0.58	−0.61	0.38	0.37	−0.69	−0.23	−0.81	−0.61	−0.15
−0.01	−0.66	−0.29	−0.02	−0.13	−0.45	−0.19	−0.27	−0.09	−0.26

Table 3
Offspring Solutions ($V^{(1)}$).

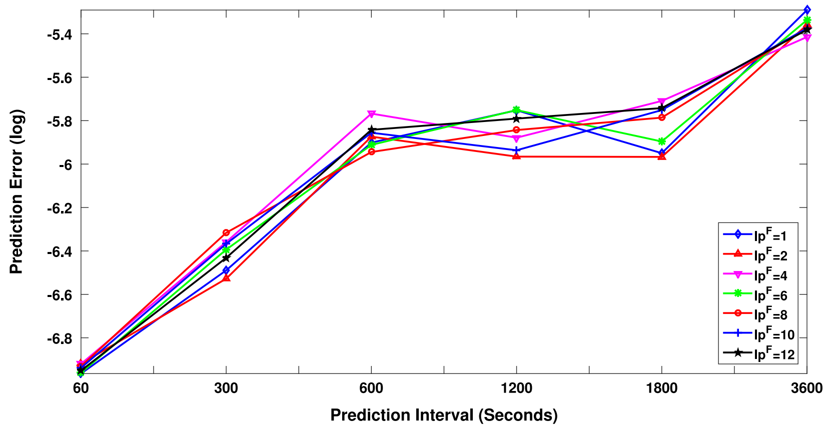
0.36	−0.58	−0.67	0.91	0.36	−0.35	−0.20	−0.96	0.59	0.50
−0.79	−0.23	−0.47	0.63	0.39	−0.88	−0.24	0.85	−0.56	−0.48
0.44	−0.58	0.79	0.91	0.37	−0.69	−0.23	0.31	0.59	−0.15
−0.01	−0.66	−0.29	0.08	0.50	−0.45	−0.19	0.87	0.15	−0.54

Table 4
Population for second iteration ($X^{(2)}$).

0.49	−0.58	−0.67	0.41	0.36	−0.76	−0.20	−0.96	−0.67	−0.12
−0.79	−0.23	−0.47	0.63	0.39	−0.88	−0.24	0.85	−0.56	−0.48
0.44	−0.58	0.79	0.91	0.37	−0.69	−0.23	0.31	0.59	−0.15
−0.07	−0.65	0.03	0.08	0.50	0.09	−0.49	0.87	0.15	−0.54

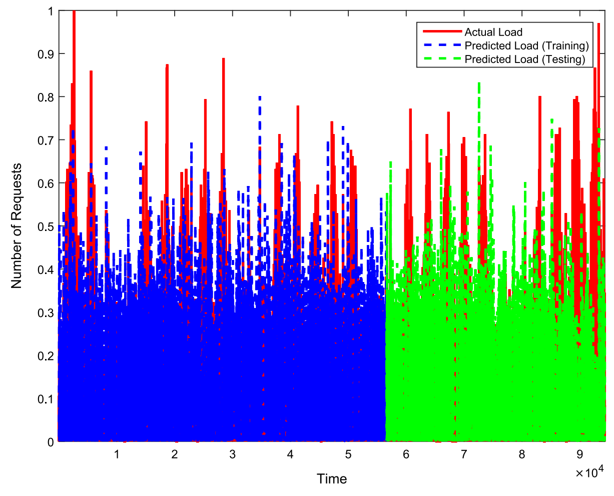


(a) NASA trace.

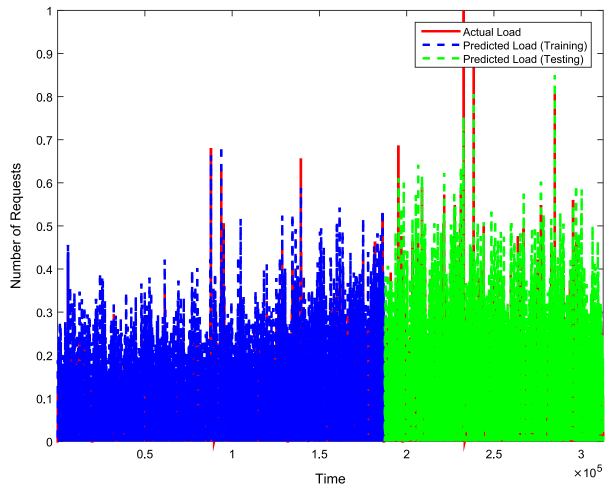


(b) Saskatchewan trace.

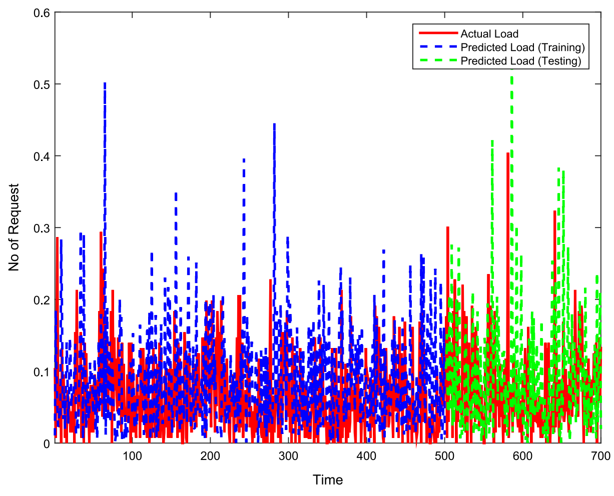
Fig. 5. Performance of prediction model on different learning periods.



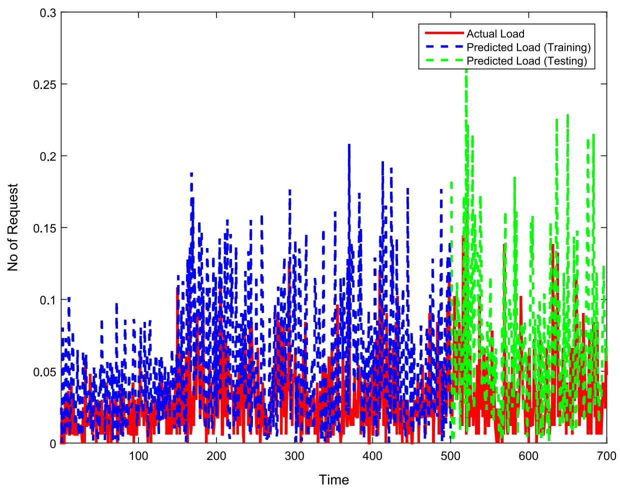
(a) NASA trace.



(b) Saskatchewan trace.



(c) NASA trace (Zoomed In)



(d) Saskatchewan trace (Zoomed In)

Fig. 6. Actual load vs predicted load (Prediction interval = 60 s).

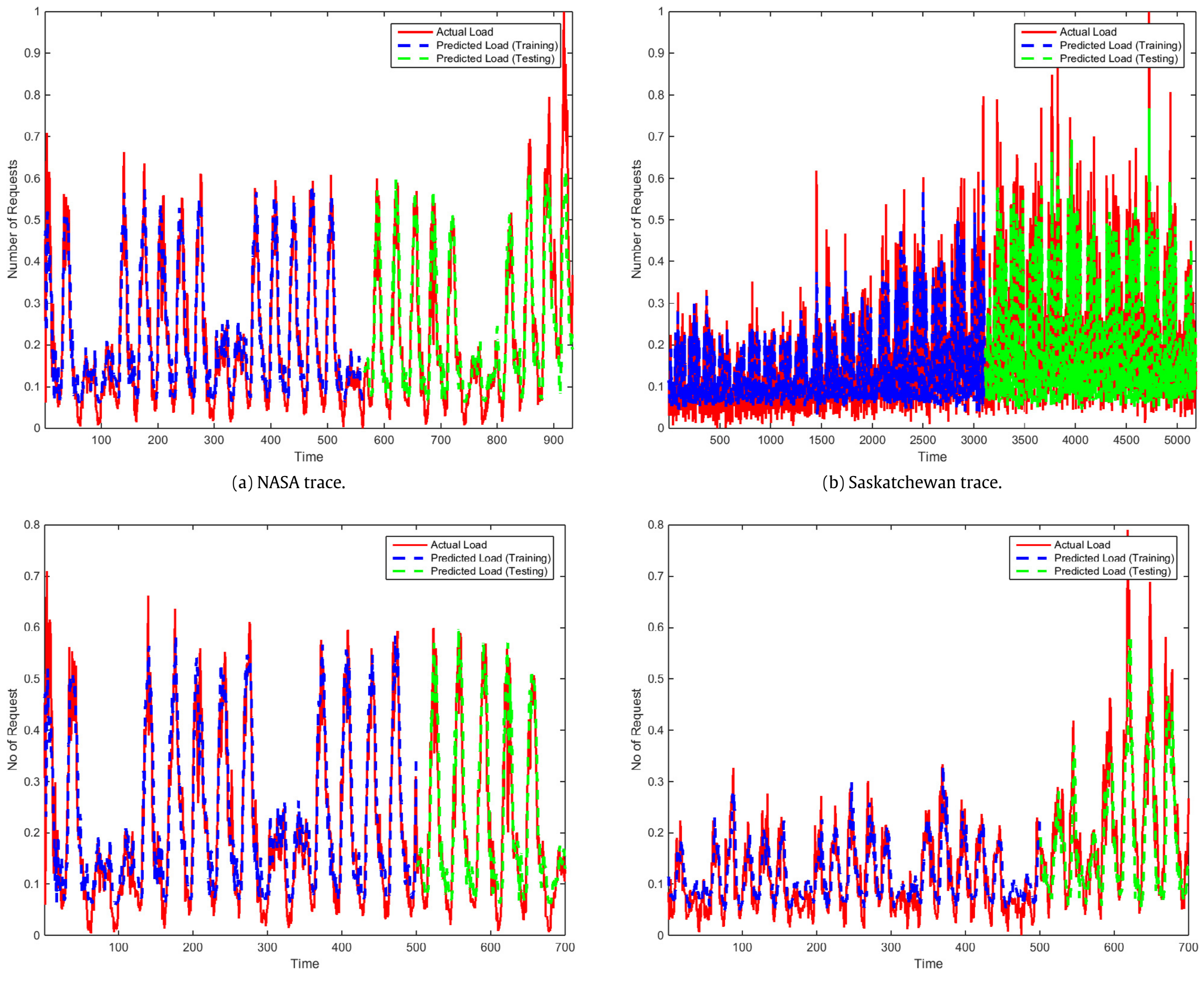
Table 5
Experimental parameters values.

Parameter	Value
Input Nodes (n)	10
Hidden Nodes (p)	7
Maximum Iterations (G_{max})	250
Training Data Size	60%
Mutation Strategy Learning Period (lp^F)	10
Crossover Rate Learning Period (lp^{CR})	10

The attributes of the data are *host, timestamp, request, HTTP reply code* and *bytes in the reply*. We performed a number of experiments with different number of input nodes ranging from 5 to 20 and the best performance was obtained with 10 input neurons. The number of hidden neurons was selected equal to $\frac{2}{3}$ of input neurons i.e. 7. The population size, mutation strategy and crossover rate learning periods were selected to be 20, 10 and 10 respectively. Fig. 5 demonstrates the effect of different learning periods on the accuracy and none of them produces best result for all experiments. The details of other parameters and their values are given in Table 5.

The detailed study was performed on the workload prediction with different prediction intervals and compared with average, maximum and back propagation based prediction models [22]. The average and maximum based approaches estimate the upcoming workload to be same as average and maximum of workload respectively. The experiments were done for prediction interval of size 1, 5, 10, 20, 30, and 60 min. In the reported experiments, 60% of data is considered to train the model while remaining 40% is used for testing of model. Each experiment was performed 10 times and an average of the obtained results are presented in this paper.

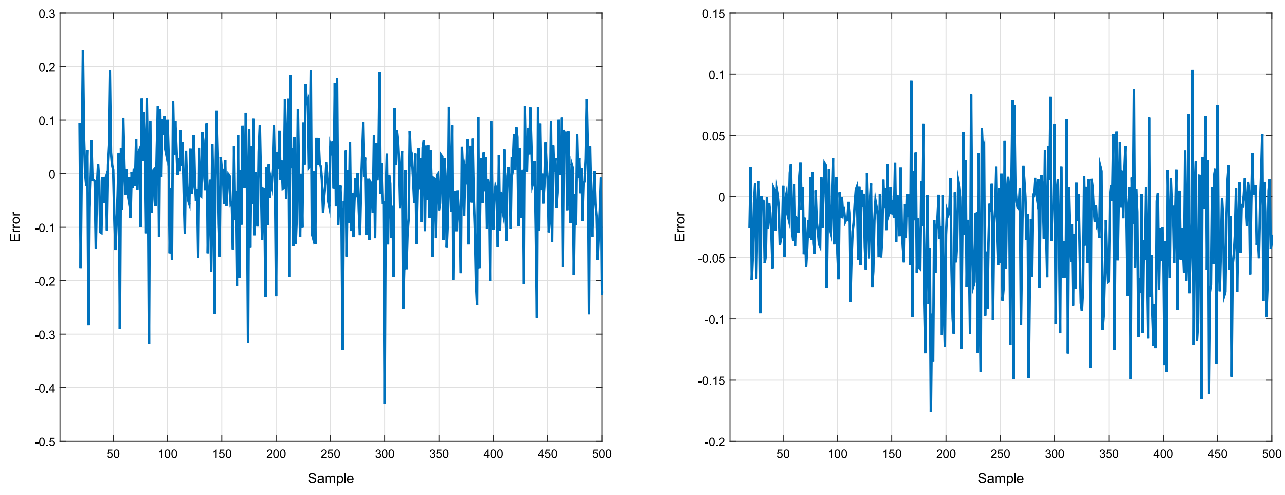
Figs. 6(a) and 6(b) show the actual and predicted workload for D_1 and D_2 respectively when prediction interval is 1 min. For visibility purpose, Figs. 6(c) and 6(d) depict first 500 samples from training data and first 200 samples from testing data. The obtained results show the close relation between actual and predicted workload. The estimated load is either equal or more than the actual load in most of the instances. However, the difference is very small and can be clearly seen from the results. The difference between actual and predicted workload ($y_i - \hat{y}_i$) for first 500 instances is also depicted in Figs. 8(a) and 8(b) for both data sets respectively. Thus the resources can be scaled dynamically according to the predicted workload. Similarly for prediction interval of 60 min, Figs. 7(a) and 7(b) show the actual and predicted load of NASA server (D_1) and Saskatchewan server (D_2) respectively. The rescaled versions of these two graphs are shown as Figs. 7(c) and 7(d) for both data sets respectively. Similar to the case of 1 min prediction interval, it is clearly evident that the predicted load is very close to actual load.



(a) NASA trace. (b) Saskatchewan trace.

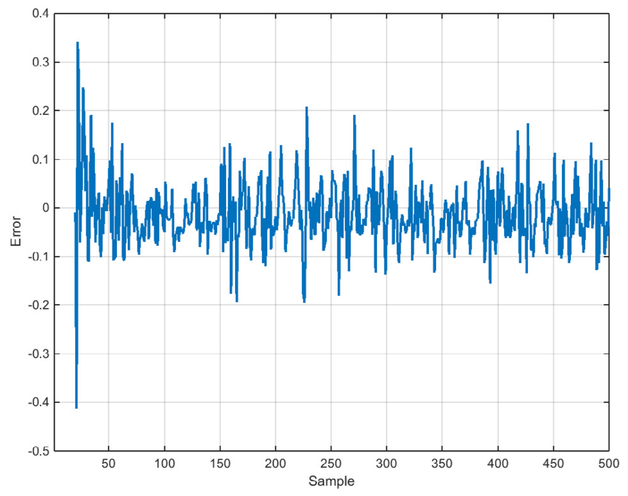
(c) NASA trace (Zoomed In) (d) Saskatchewan Trace (Zoomed In)

Fig. 7. Actual load vs predicted load (Prediction interval = 60 min).

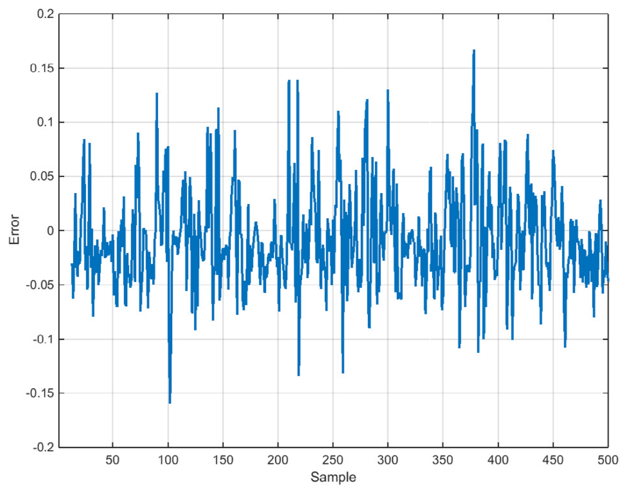


(a) NASA trace. (b) Saskatchewan trace.

Fig. 8. Prediction error $(y_i - \hat{y}_i)$ (Prediction interval = 60 s).

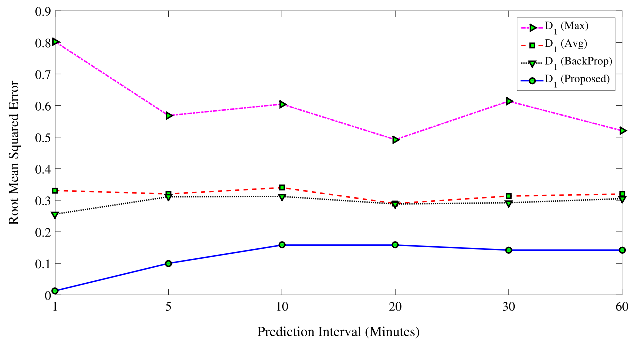


(a) NASA trace.

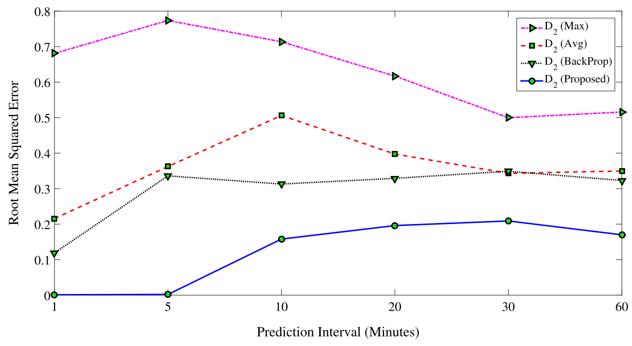


(b) Saskatchewan trace.

Fig. 9. Prediction error $(y_i - \hat{y}_i)$ (Prediction interval = 60 min).



(a) NASA trace.



(b) Saskatchewan trace.

Fig. 10. Root mean squared error of average, maximum, back propagation based prediction models and proposed approach.

Table 6
Average, maximum, back propagation vs proposed model (RMSE)

Prediction Interval (min)	Root mean squared error							
	Maximum		Average		Back propagation		Proposed	
	D_1	D_2	D_1	D_2	D_1	D_2	D_1	D_2
1	0.802	0.681	0.330	0.215	0.256	0.119	0.013	0.001
5	0.568	0.773	0.320	0.362	0.311	0.336	0.100	0.002
10	0.604	0.713	0.339	0.506	0.312	0.313	0.158	0.158
20	0.492	0.617	0.289	0.397	0.288	0.329	0.158	0.196
30	0.613	0.499	0.313	0.343	0.292	0.349	0.142	0.209
60	0.520	0.515	0.319	0.349	0.305	0.323	0.142	0.170

The difference in first 500 estimations for 60 min prediction interval are also shown in Fig. 9 for both data sets. The obtained root mean squared errors for D_1 and D_2 using average, maximum and back propagation based models and proposed model are shown in Table 6 for each prediction interval. The results show a substantial improvement in root mean squared error. The proposed approach outperforms other three methods in all experiments. The maximum shift over well known back propagation based model is achieved for predictions of data set D_2 when prediction interval is 5 min. In this case the RMSE produced by proposed model is 0.002 while back propagation based model produces 0.336. The same results are graphically rendered in Figs. 10(a) and 10(b) for both data sets D_1 and D_2 respectively. On comparing the results with average and maximum based approaches, we found an improvement up to 25 and 61 times for NASA trace, and 125 and 680 times for Saskatchewan trace respectively. This improvement defines the reduction in root mean squared error. The results present satisfying advancement in accuracy of predictions using the proposed method over average, maximum and back propagation based prediction model.

Table 7 shows the number of iterations elapsed in training for each prediction window size using back propagation and self adaptive differential evolution algorithms. The maximum iterations were set to 250 for each experiment. Back propagation took complete 250 iterations for all experiments while self adaptive differential evolution based model elapsed less than 80 iterations in each case. The results show that the convergence is achieved in less number of iterations comparatively to back propagation. However, SaDE consumed longer time in training as shown in Table 8. It is due to the fact that the evolutionary approach works

Table 7
Iterations elapsed in training (back propagation vs SaDE)

Prediction Interval (min)	Iterations Elapsed in Training			
	Back propagation		SaDE	
	D1	D2	D1	D2
1	250	250	47	61
5	250	250	39	77
10	250	250	51	51
20	250	250	47	51
30	250	250	26	42
60	250	250	26	21

Table 8
Elapsed time in training (seconds)

PWS (min)	SaDE		Back Propagation	
	NASA	Saskatchewan	NASA	Saskatchewan
1	290.823	1949.648	121.368	661.690
5	48.539	498.452	24.112	132.666
10	31.700	168.668	12.144	64.807
20	14.759	85.497	6.354	33.376
30	5.507	46.741	4.274	21.429
60	3.502	12.238	2.908	10.733

with a number of solutions. The obtained results are promising in predicting workload on cloud data centers and proposed approach outperforms back propagation based neural network and other two models in prediction accuracy.

5. Conclusions

The accurate workload prediction can be used effectively to reduce the number of SLA violations and operational cost of the data centers. The obtained results are convincing and encourage us to explore the area further. The model achieved notable transformation in root mean squared error over models based on average, maximum and back propagation. The proposed approach resulted in the minimum prediction error as 0.013 and 0.001 for D_1 and D_2 respectively. While back propagation generated minimum error as 0.265 and 0.119 for D_1 and D_2 respectively. The utmost shift attained by proposed model over back propagation based model is 168 times in the predictions of Saskatchewan server traces when prediction interval is 5 min. The improvement in accuracy is achieved due to the fact that evolutionary approach explores the solution space in multiple directions using a number of solutions. This avoids the risk of being trapped into local optima as compared to gradient based learning algorithms. Back propagation based model consumed 250 iterations while SaDE based prediction model elapsed less than 80 iterations. Based on the results, we conclude that the proposed model outperforms the back propagation based prediction model. An effective workload estimation model in the cloud data centers will improve the efficiency along with reduction in operational cost. The aim of future work is to improve the prediction accuracy along with use of virtual machine placement and migration scheme to reduce the SLA violations. In addition, a prediction model based on different attributes of the workload can be explored further.

Acknowledgments

The authors would like to thank Ministry of Electronics & Information Technology (MeitY), Government of India for providing financial support to carry out this work under Visvesvaraya Ph.D. Scheme for Electronics & IT. Authors are also thankful to the anonymous reviewers for their constructive comments and suggestions to improve the manuscript.

References

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Gener. Comput. Syst. 25 (6) (2009) 599–616.
[2] S.S. Manvi, G. Krishna Shyam, Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey, J. Netw. Comput. Appl. 41 (2014) 424–440.
[3] N. Roy, A. Dubey, A. Gokhale, Efficient autoscaling in the cloud using predictive models for workload forecasting, in: 2011 IEEE 4th International Conference on Cloud Computing, IEEE, 2011, pp. 500–507.
[4] K. Cetinski, M.B. Juric, AME-WPC: Advanced model for efficient workload prediction in the cloud, J. Netw. Comput. Appl. 55 (55) (2015) 191–201.
[5] Lingyun Yang, I. Foster, J.M. Schopf, Homeostatic and tendency-based CPU load predictions, in: Proceedings International Parallel and Distributed Processing Symposium, IEEE Comput. Soc., 2014, p. 9.
[6] S.-R. Kuang, K.-Y. Wu, B.-C. Ke, J.-H. Yeh, H.-Y. Jheng, Efficient architecture and hardware implementation of hybrid fuzzy-Kalman filter for workload prediction, Integration VLSI J. 47 (4) (2014) 408–416.
[7] C. Liu, C. Liu, Y. Shang, S. Chen, B. Cheng, J. Chen, An adaptive prediction approach based on workload pattern discrimination in the cloud, J. Netw. Comput. Appl. 80 (2017) 35–44.
[8] T.H. Li, A hierarchical framework for modeling and forecasting web server workload, J. Amer. Statist. Assoc. 100 (471) (2005) 748–763.
[9] D. Ardagna, S. Casolari, M. Colajanni, B. Panicucci, Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems, J. Parallel Distrib. Comput. 72 (6) (2012) 796–808.
[10] P.S. Kalekar, Time series forecasting using Holt–Winters exponential smoothing. Technical Report. 2004.
[11] Y.S. Sun, Y.-F. Chen, M.C. Chen, A workload analysis of live event broadcast service in cloud, Procedia Comput. Sci. 19 (2013) 1028–1033.
[12] A. Khan, Xifeng Yan, Shu Tao, N. Anerousis, Workload characterization and prediction in the cloud: A multiple time series approach, in: 2012 IEEE Network Operations and Management Symposium, IEEE, 2012, pp. 1287–1294.
[13] T. Vercauteren, P. Aggarwal, X. Wang, T.-H. Li, Hierarchical forecasting of web server workload using sequential Monte Carlo training, IEEE Trans. Signal Process. 55 (4) (2007) 1286–1297.
[14] L. Cao, Support vector machines experts for time series forecasting, Neurocom puting 51 (2003) 321–339.
[15] T. Ban, R. Zhang, S. Pang, A. Sarrafzadeh, D. Inoue, Referential kNN Regression for Financial Time Series Forecasting, Springer, Berlin, Heidelberg, 2013, pp. 601–608.
[16] A. Eddahech, S. Chtourou, M. Chtourou, Hierarchical neural networks based prediction and control of dynamic reconfiguration for multilevel embedded systems, J. Syst. Archit. 59 (1) (2013) 48–59.
[17] J. Yang, C. Liu, Y. Shang, B. Cheng, Z. Mao, C. Liu, et al., A cost-aware autoscaling approach using the workload prediction in service clouds, Inf. Syst. Front. 16 (1) (2014) 7–18.
[18] R. Hu, J. Jiang, G. Liu, L. Wang, Efficient resources provisioning based on load forecasting in cloud, Sci. World J. (2014) 3212–3231.
[19] V.R. Messias, J.C. Estrella, R. Ehlers, M.J. Santana, R.C. Santana, S. Reiff-Marganiec, Combining time series prediction models using genetic algorithm to autoscaling web applications hosted in the cloud infrastructure, Neural Comput. Appl. 27 (8) (2016) 2383–2406.
[20] S. Islam, J. Keung, K. Lee, A. Liu, Empirical prediction models for adaptive resource provisioning in the cloud, Future Gener. Comput. Syst. 28 (1) (2012) 155–162.
[21] V.G. Tran, V. Debusschere, S. Bacha, Hourly server workload forecasting up to 168 h ahead using Seasonal ARIMA model, in: 2012 IEEE International Conference on Industrial Technology, IEEE, 2012, pp. 1127–1131.
[22] J.J. Prevost, K. Nagothu, B. Kelley, M. Jamshidi, Prediction of cloud data center networks loads using stochastic and neural models, in: 2011 6th International Conference on System of Systems Engineering, IEEE, 2011, pp. 276–281.
[23] E. Caron, F. Desprez, A. Muresan, Pattern matching based forecast of non-periodic repetitive behavior for cloud clients, J. Grid Comput. 9 (1) (2011) 49–64.
[24] Z. Chen, Y. Zhu, Y. Di, S. Feng, Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network, Comput. Intell. Neurosci. (2015) 1–14.
[25] Zhenhuan Gong, Xiaohui Gu, J. Wilkes, PRESS: PRedictive Elastic ReSource Scaling for cloud systems, in: 2010 International Conference on Network and Service Management., IEEE, 2010, pp. 9–16.
[26] Y.-C. Chang, R.-S. Chang, F.-W. Chuang, A Predictive Method for Workload Forecasting in the Cloud Environment, Springer, Dordrecht, 2014, pp. 577–585.
[27] Y. Lu, J. Panneerselvam, L. Liu, Y. Wu, RVLBPNN: A workload forecasting model for smart cloud computing, Sci. Program. (2016) 2016.
[28] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (4) (1943) 115–133.

[29] B. Hall, B. Hallgrimsson, Srickberger's Evolution, fourth ed., Jones & Bartlett, 2008.
[30] A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: IEEE Congress on Evolutionary Computation, IEEE, 2005, pp. 1785–1791.
[31] W. Liu, H. Song, J.J. Liang, B. Qu, A.K. Qin, Neural Network Based on Self-adaptive Differential Evolution for Ultra-Short-Term Power Load Forecasting, Springer, Cham, 2014, pp. 403–412.
[32] A. Deb, J.S. Roy, B. Gupta, Performance comparison of differential evolution, particle swarm optimization and genetic algorithm in the design of circularly polarized microstrip antennas, IEEE Trans. Antennas and Propagation 62 (8) (2014) 3920–3928.
[33] C. Worasucheep, P. Chongstitvatana, A Multi-strategy Differential Evolution Algorithm for Financial Prediction with Single Multiplicative Neuron, Springer, Berlin, Heidelberg, 2009, pp. 122–130.
[34] N. Karaboga, B. Cetinkaya, Performance Comparison of Genetic and Differential Evolution Algorithms for Digital FIR Filter Design, Springer, Berlin, Heidelberg, 2004.
[35] Hatem Abdul-Kader, M.A. Salam, Evaluation of differential evolution and particle swarm optimization algorithms at training of neural network for stock prediction, Int. Arab J. e-Technol. 2 (3) (2012) 145–150.
[36] X. Dong, S. Liu, T. Tao, S. Li, K. Xin, A comparative study of differential evolution and genetic algorithms for optimizing the design of water distribution systems, J. Zhejiang Univ. Sci. A 13 (9) (2012) 674–686.
[37] E. Mezura-Montes, CAC. Coello, An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, Int. J. Gen. Syst. 37 (4) (2008) 443–473.
[38] Traces available in the Internet Traffic Archive. http://ita.ee.lbl.gov/html/traces.html.

Jitendra Kumar is currently working towards Ph.D. in the department of computer applications of National Institute of Technology Kurukshetra, India. He received M.Sc. and M.Tech both in computer science from Dayalbagh Educational Institute Agra, India respectively in 2011 & 2013. He was awarded with Directors medal for fetching first rank in M.Sc. at university level. He has experience of more than 2 years in industry and academia. He is actively engaged in research activities and has published 6 research articles in international conferences of repute. His current research interests include cloud resource management, machine intelligence, data analytics, optimization, high performance computing.

Ashutosh Kumar Singh is working as a Professor and Head in the department of computer applications, National Institute of Technology; Kurukshetra, India. He has more than 15 years research and teaching experience in various university systems of the India, UK, Malaysia and Australia. Dr. Singh has obtained Ph.D. degree in Electronics Engineering from Indian Institute of Technology, BHU, India, Post Doc from Department of Computer Science, University of Bristol, UK and Charted Engineer from UK.

His research area includes Verification, Synthesis, Design and Testing of Digital Circuits, Cloud Computing, Machine Learning, Security, Big Data. He has published more than 135 research papers till now in different journals, conferences and news magazines and in these areas. He is the co-author of six books which includes "Web Spam Detection Application using Neural Network", "Digital Systems Fundamentals" and "Computer System Organization & Architecture". He has worked as principal investigator for four sponsored research projects and was a key member on a project from EPSRC (UK) "Logic Verification and Synthesis in New Framework".

Dr. Singh has delivered the invited talks and presented research papers in several countries including Australia, UK, South Korea, China, Thailand, Indonesia, India and USA. He had been entitled for the awards such as Merit Award-03 (Institute of Engineers), Best Poster Presenter-99 in 86th Indian Science Congress held in Chennai, INDIA, Best Paper Presenter of NSC'99 INDIA and Bintulu Development Authority Best Postgraduate Research Paper Award for 2010, 2011, 2012.

He has shared his valuable experience as guest editor/editorial board member of several international journal and organizing chair of few international conferences. He is involved in reviewing process in different journals and conferences such as; IEEE transaction of computer, IET, IEEE conference on ITC, ADCOM etc.