Information Sciences 543 (2021) 345–366

Contents lists available at ScienceDirect

Information Sciences

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

Self directed learning based workload forecasting model for cloud resource management

Jitendra Kumar [a,*], Ashutosh Kumar Singh [b], Rajkumar Buyya [c]

[a] Department of Computer Applications, National Institute of Technology Tiruchirappalli, India
[b] Department of Computer Applications, National Institute of Technology Kurukshetra, India
[c] Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, VIC 3010, Australia

ARTICLE INFO

Article history:
Received 26 February 2019
Received in revised form 6 July 2020
Accepted 8 July 2020
Available online 25 July 2020

Keywords:
Workload prediction
Resource demand
Blackhole algorithm
Neural network
Optimization
Statistical analysis

ABSTRACT

Workload prediction plays a vital role in intelligent resource scaling and load balancing that maximize the economic growth of cloud service providers as well as users' quality of experience (QoE). Numerous approaches have been discovered to estimate the future workload and machine learning is being widely used to improve the forecast accuracy. This paper presents a self directed workload forecasting method (SDWF) that captures the forecasting error trend by computing the deviation in recent forecasts and applies it to enhance the accuracy of further predictions. The model utilizes an improved heuristic approach based on the blackhole phenomena for the training of neurons. The efficacy of the proposed method is evaluated over six different real world data traces. The accuracy of the model is compared with existing models that use different state-of-art approaches including deep learning, differential evolution and back propagation. The maximum relative reduction in mean squared forecast error is observed up to 99.99% compared with existing methods. In addition, the statistical analysis is also carried out using Friedman and Wilcoxon signed rank tests to validate the efficacy of the proposed forecasting model.

1. Introduction

Cloud computing allows a service provider to share the resources and applications among users. The key characteristics of a cloud system are elasticity, on-demand services, accessibility, and pay-as-you-use model. A pool of configurable computing resources and high-level services are delivered to the users through virtualization in the form of virtual resources or virtual machines over the Internet. In general, a cloud system is equipped with a set of high-end data centers to deliver on-demand services. The high non-linearity in the workload affects the performance of a cloud system and leads towards various issues such as low resource utilization, high power consumption, and inconsistent quality of services (QoS) [1]. For instance, a study of IBM reported 17.76% and 77.93% average CPU and memory utilization respectively [2]. A similar survey of Google states that the CPU and memory usage of a Google cluster could not exceed beyond 60% and 50% respectively [3]. The inefficient functioning of a data center increases its operational cost and fiscal loss of service providers. Also, low resource usage results in high operational costs due to excessive use of electricity. The increase in electricity consumption also impacts the carbon footprints, and it must be minimized because an ideal active machine consumes over half of the peak power consumption [4]. According to Environmental Impact Assessment (EIA), data centers used 35 TWh electricity in 2015 and is expected to

* Corresponding author.
E-mail addresses: jitendra@nitt.edu (J. Kumar), ashutosh@nitkkr.ac.in (A.K. Singh), rbuyya@unimelb.edu.au (R. Buyya).

https://doi.org/10.1016/j.ins.2020.07.012
0020-0255/

increase up to 95 TWh by 2040 [5]. The electricity consumption in cloud infrastructure would place it sixth in a ranking of power consumption per country.

An intelligent resource provisioning mechanism can effectively address the presence of high variability in the workloads. A resource provisioning method allocates the resources to the applications according to their demands. It must avoid the occurrence of over-provisioning or under-provisioning i.e. assigning more or fewer resources as per their needs. The resource provisioning methods can be classified into two categories, namely reactive and proactive. The reactive methods allocate the resources after arrival of demands, whereas the proactive methods allocate the resources in advance. The proactive approaches provide the resources based on an estimation of workloads that can be predicted using historical information. The anticipated information can be effectively used for intelligent dynamic resource scaling [6]. Several methods have been proposed to estimate the future workload using different learning methods, including neural network, genetic algorithm, and regression. However, the accurate forecasting is a complex task, especially in the presence of inconsistent and non-linear workloads. It has been a challenge for any forecasting model to incorporate real time accurate forecasting that complicates a prediction system. Since it has become mandatory for a cloud service provider to achieve better scalability, response time, QoS, and other parameters to sustain in today's competitive market [7]. Therefore, this paper presents a self directed workload forecasting approach to improve the forecast quality that can help a cloud system in improving its service quality.

1.1. Our contributions

This paper proposes a novel self-directed workload forecasting method to estimate the future workload on cloud servers. The proposed framework is capable of learning from its past forecasts to improve future estimates. The primary contributions of the work are twofold. First, the forecast error feedback is introduced that enables the model to learn from its recent forecasting pattern. The forecasting module receives the feedback to leverage it in the next forecast. The model captures the error in recent l forecasts and computes the average deviation. Second, a population based metaheuristic optimization algorithm, i.e., blackhole algorithm is improved for better learning of network weights to achieve more accurate forecasts. The new learning algorithm organizes the population into multiple clusters or subpopulations. Also, the local and global best information is incorporated in the process of generation of new solutions as opposed to its standard algorithm that considers the global best information only to generate new solutions. The incorporation of the local best information has been found useful in preserving the diversity of the population that prevents premature convergence.

The rest of the paper is organized as follows: Section 2 discusses the recent key contributions in the workload prediction approaches in the machine learning domain. Section 3 describes the proposed model in depth followed by results and discussion in Section 4. Finally, the paper is wrapped up with conclusive remarks in Section 6.

2. Related work

The prediction has a wide range of applications such as web service recommendation, market and, wind power generation. It also finds the significant applicability in cloud resource management by estimating the expected workload on the servers [8–10]. The workload prediction methods are broadly classified into table-driven, control theory, queuing theory and machine learning methods [11]. Among the mentioned approaches, machine learning has been explored widely for workload estimation and this paper also presents a machine learning based method. Therefore, the recent significant contributions in workload prediction using machine learning only are discussed.

A virtual machine workload prediction was proposed in [12] which uses the estimated information to determine whether an application is CPU intensive and/or memory intensive to configure the resources accordingly. An evolutionary neural network was used for workload prediction [13]. The approach implements particle swarm optimization, differential evolution, and covariance matrix adaptation evolutionary strategy learning algorithms and compares their performance. Kumar and Singh also used a neural network based prediction approach to estimate the future load on servers [14]. The model was trained using adaptive differential evolution to reduce the overhead of the parameter tuning in the evolutionary algorithms. The concept of sliding window has been used with neural network and linear regression for estimating the future workloads [15]. Similarly, various neural network based prediction approaches have been proposed to improve the forecast accuracy including [16–18].

Valter et al. developed a forecasting model that incorporates multiple time series forecasting approaches [19]. In this model, each time series forecasting method makes its predictions based on its extracted pattern and each method's prediction is weighted to compute the final forecast. The authors used genetic algorithm to generate an effective weighted model. A predictive framework for resource provisioning to optimize the energy consumption was developed in [20] that predicts the VMs along with their resource demands expected to arrive in the future. The method also provides the information about required physical resources to fulfill the future demands with optimized use of energy. The predictive approach involves the use of clustering and Wiener filter. Shen and Chen studied the different servers' resource utilization and found that they are misaligned with time [21]. They developed a set of algorithms to refine the utilization patterns to reduce the resource over-provisioning. Genetic algorithm based workload predictive resource management was proposed in [22] which improves the average utilization and energy consumption. The method is a combination of prediction and placement

approaches. Liu et al. carried out a comparative analysis on extreme learning machines (ELMs), online sequential ELMs (OS-ELM), support vector machines (SVM), and OS-SVM for the job failure prediction in the cloud system [23]. The work performs three tests on offline, online, and parameter optimization. Zhang et al. proposed the intelligent workload factoring to achieve proactive workload management [24]. It uses various components of applications to categorize the workloads into two different categories called base and flash crowd. It also detects the items to factor the incoming requests in the context of data and volume. The linear regression based method has been used in optimizing resource scaling decisions [25]. The approach keeps the resource scaling cost low and meets the service level agreements. Pulido et al. developed a hybrid forecasting approach that uses an ensemble of neural networks trained using particle swarm optimization [26]. The method aggregated the individual forecast responses using type-2 fuzzy logic integration to produce the final forecast value. The fuzzy integrators in the neural network ensemble were optimized using genetic algorithm [27]. The approach optimized the fuzzy inference system's membership function parameters in each integrator. A hybrid prediction scheme called 'ClusFuDE' was developed for low dimensional numerical data trace forecasting that categorized data using an improved clustering method and suboptimal solutions were optimized using differential evolution [28]. The authors used fuzzy logical relationships to estimate the predicted values that were defuzzified to compute the real value of the forecast.

A resource allocation and power management framework is developed using deep learning in [29] which includes a forecaster to estimate workload for power manager. The long short term memory (LSTM) recurrent neural network has been used as an underlying methodology to develop the forecaster module. The power manager initiates further actions based on the estimations and the current state information. Qiu et al. also proposed a predictive method using deep learning architectures to estimate VM workloads [30]. The model used a set of restricted Boltzmann machines arranged in a layered approach along with a regression layer. A workload prediction mechanism based on LSTM networks had also been explored in [31] which used four LSTM units to improve the quality of forecasts. A deep reinforcement learning based resource management scheme was proposed in [32]. The resource manager was composed of monitor, allocator, and controller devices which were responsible for resource utilization information gathering, mapping of applications to the resource pool, and resource configuration negotiation respectively. Patel and Mishra also used deep learning to develop a prediction method that uses the past workload information to compute the correlation among VMs and predicts the future workload information accordingly [33]. An efficient workload prediction model based on deep learning was presented in [34]. The model converted the weight vectors into canonical polyadic decomposition to compress the model attributes. In addition, the work also proposed a learning methodology based on backpropagation for the training of the auto encoder's parameters.

A resource management scheme utilizing forecasting and skewness was introduced in [35]. The approach minimizes the skewness to combine the different categories of workloads which improves resource utilization. The run-length encoding based prediction scheme for processor power management had been presented in [36]. The approach was energy efficient and efficaciously addressed the repetitious behavior of workloads. A prediction method based on sequential pattern mining was presented in [37]. The correlation between resource variables was considered in pattern extraction of applications' behavior and these patterns were used for workload forecasting on the cloud server. Further, a prediction approach based on episode mining with online learning capability was proposed in [38]. The learning method was inspired by the different categories of human memory called long term and short term memory. The long term memory was responsible for storing the episodes of application behavior a in long period while the short term memory stores the new behavior of applications which correspond to the online learning.

Kaur et al. proposed an intelligent regression ensemble approach (REAP) to improve the forecast accuracy that combines the resource usage prediction approaches with feature selection [39]. The prediction scheme that use stochastic configuration networks in combination with Savitzky-Golay filter and wavelet decomposition is proposed in [40]. The approach applies the SG filter to smooth data trace which is decomposed into various parts using wavelet decomposition. The decomposed parts of the data trace are used to extract the statistical features of trend and detailed trace components using an established prediction scheme. A similar approach was proposed that uses noise filtering and data frequency representation named SGW-SCN to estimate future workload information [9]. Lin et al. proposed an artificial neural network based forecasting scheme to model the power consumption of a data center [41]. The power consumption and system performance characteristics of different components including CPU, primary and secondary memory are analyzed. The authors proposed the power consumption models based on back propagation, Elman and LSTM networks. A prediction scheme that first analyzes the dependence in a large scale system and prepares two different time series models based on day and time is proposed in [42]. An improved LSTM model is proposed to forecast the future workload using two dimensional time series information.

These methods use prediction mechanisms which are usually fit for a specific pattern of workloads and fails in handling the real world traces where the pattern changes rapidly over time. In such cases, the resources remain over-provisioned or under-provisioned. Therefore, a scheme that can adapt to sudden changes becomes more useful. In this context, an adaptive learning based predictive framework is introduced in [43] that learns the operators as per workload characteristics. A workload prediction framework 'CloudInsight' was developed using a set of predictors [44]. It combines 8 different prediction methods from machine learning, time series, and regression classes to improve the accuracy of forecasts. Zhong et al. predicted the workload sequences using weighted wavelet support vector machines [45]. The approach used wavelet transform and support vector machines to improve the forecast accuracy. The model parameters were optimized using the particle swarm optimization based approach. This paper contributes by developing a workload prediction approach that learns from the forecasting errors in the past and tune itself accordingly. It also improves one of the metaheuristic optimization algorithms which is inspired by the blackhole phenomena to achieve more accurate weights of the network.

3. Workload modeling and estimation

A cloud data center is equipped with several physical machines or servers to offer on-demand services to the customers. The applications are hosted in the form of virtual machines that are placed on one or more servers. The resource utilization and the current state of the data center are monitored over time using a device called *resource manager*. The task of the resource manager is to assign the resources to the applications in an efficient manner, and predictive models help in achieving the same by providing an estimation of future workload. The predictive resource assignment schemes improve the economic growth of the service provider by maximizing the utilization and reducing electricity consumption.

The workload predictor anticipates the future trends of the number of requests, the demands or utilization pattern of resources based on the previous information. The complete process flow of the proposed method is shown in Fig. 1. First, the historical workload data traces are extracted and preprocessed using aggregation, differencing, and rescaling. Further, a part of preprocessed data is used to learn the network weights, and the remaining data is used to evaluate the accuracy of the trained prediction model. The forecast accuracy is measured using mean squared error i.e. squared difference between actual and predicted workload. The forecast error computation is shown during model training and deployment in Fig. 1. The actual workloads at different time instances (training and deployment) are represented with the dotted blue line and solid black line respectively, that are used to compute the forecast accuracy. The individual components of the prediction model are discussed in-depth in following sub sections.

The prediction model computes the future workload estimations using a multi-layer neural network of architecture with $n-p-q$ neurons, where n, p and q represent the number of input, hidden and output neurons correspondingly. The workload at previous n instances of time becomes the input of model to forecast the workload at next time instance e.g. $Y = \{y_1, y_2, \ldots, y_t\}$ is workload trace, $\{y_t, y_{t-1}, \ldots, y_{t-n}\}$ becomes model input to forecast the estimated workload at time instance $t+1$ ($\hat{y}_{t+1}$). Further, the data is split into two different parts referred to as training and testing data, where training data is used during the network weights learning process while the testing data is used to evaluate the accuracy of the forecasts. The accuracy of the method is evaluated using mean squared error (MSE) (see Eq. (10)), where y_t and $\hat{y}_t$ are the actual and predicted workloads respectively at time instance t, while m is the total number of samples under consideration. The operational summary of the prediction method is given in Algorithm 1.

3.1. Self directed learning

Let $Y = \{y_1, y_2, \ldots, y_t\}$ is a set of given workloads, where y_t is the load on the server at time t. The forecaster (f_{pred}) analyses previous n workloads and forecasts the next possible workload value, denoted as $\hat{y}_{t+1}$. The error in forecast (e_{t+1}) is computed as the difference between actual and predicted workload. Fig. 2 differentiates both simple and self directing forecasting approaches. The simple forecasting method analyzes the previous t workload instances to anticipate the future workload whereas the self directed forecaster includes the error feedback obtained from previous l forecasts. The presence of forecast errors can be modeled by applying δ operation that observes the arithmetic difference between actual and predicted workload as shown in Eq. (1). The forecast error trend (μ_e) is computed using Ψ operation that calculates the average of previous forecasts of error feedback window length as shown in Eq. (2). The self directed learning leverage this information (μ_e) to redirect the predictions towards actual workload values as given in Eq. (3).

$$\begin{aligned} e_{t+1} &= \delta_{t+1}\left(y_{t+1}, \hat{y}_{t+1}\right) \\ &= y_{t+1} - \hat{y}_{t+1} \end{aligned} \tag{1}$$

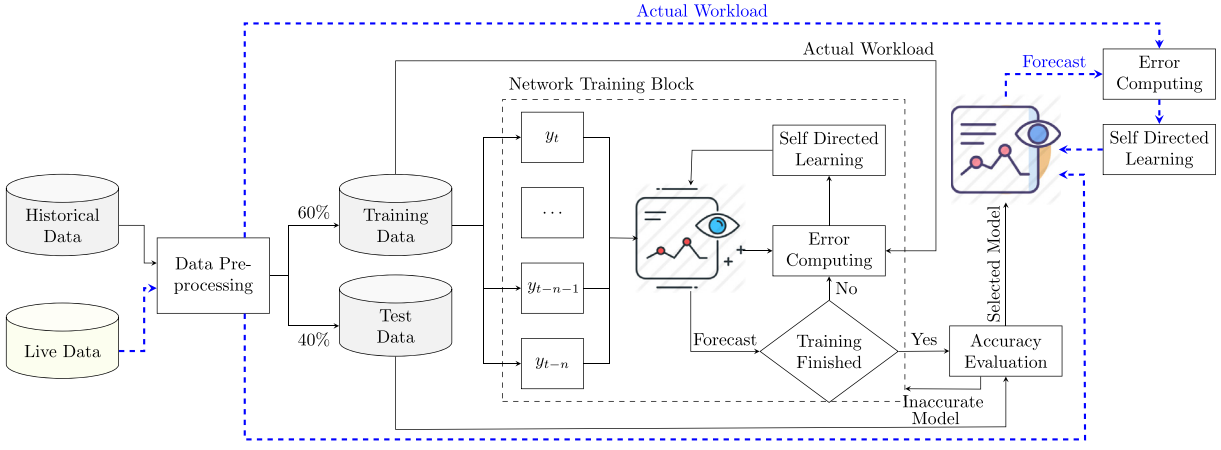


Fig. 1. Prediction model workflow.

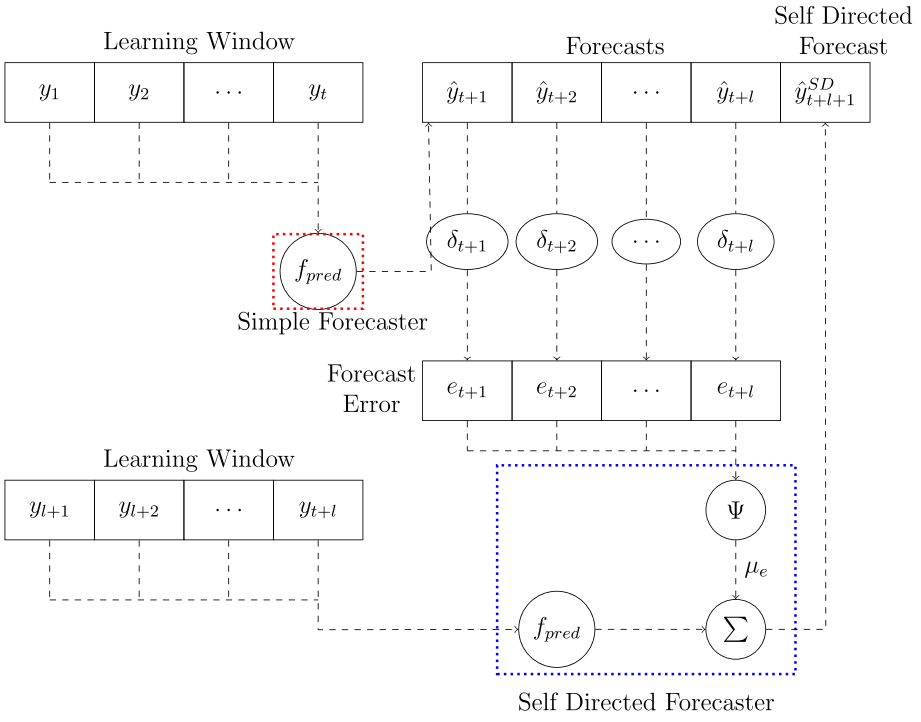


Fig. 2. Self directed learning.

$$\begin{aligned}\mu_e &= \Psi(\delta_1, \delta_2, \ldots, \delta_l) \\ &= \tfrac{1}{l}\sum_{i=1}^{l}\delta_i(y_i, \hat{y}_i)\end{aligned} \tag{2}$$

$$\hat{y}^{SD}_{t+l+1} = f_{pred}(y_{t+l}, y_{t+l-1}, \ldots, y_{l+1}) + \Psi(e_{t+l}, e_{t+l-1}, \ldots, e_{t+1}) \tag{3}$$

For workload instances Y={0.15, 0.97, 0.95, 0.48, 0.80, 0.14, 0.42, 0.91, 0.79, 0.95} an arbitrary forecaster predicts $\widehat{Y}$ = {0.0, 1.17, 1.17, 0.83, 1.06, 0.59, 0.79, 1.14, 1.05, 1.17} and its corresponding self directed forecaster predicts $\widehat{Y}^{SD}$ = {0.0, 1.17, 1.17, 0.75, 0.83, 0.42, 0.60, 0.97, 0.87, 1.06} by incorporating the error in previous three forecasts. The mean squared error for both directed and non-directed forecasts are 0.03 and 0.08 respectively.

3.2. SDWF method

The predictive model employs a population based multiple blackhole algorithm inspired by [46] that follows the blackhole phenomena of nature. In standard blackhole (SB) optimization algorithm, the stars or candidate solutions move rapidly towards the best solution that reduces the population diversity and exploration capacity of the stars that may lead the approach towards finding the sub-optimal solution [47]. The SDWF avoids the premature convergence by organizing the stars into c clusters or sub populations and includes the local best solution in position update procedure. Consider, $S = \{s_1, s_2, \ldots, s_P\}$ is a set of P random stars uniformly distributed in the search space and initialized using Eq. (4). Here, $lb_j = -1$ and $ub_j = 1$ define the lower and upper bounds of search space in j^{th} dimension and $r \in [0, 1]$ is a random number. The S is organized into $k = \{1, 2, \ldots, c\}$ clusters each of size P/c and every individual (s_i^k) is evaluated over training data by computing the mean squared error of forecasts. The best solution of each k^{th} cluster is considered as its local blackhole (β_l^k) and best among local blackholes is designated as global blackhole (β_g). Next, the position of stars get updated to find the optimal solution. Unlike SB the SDWF's position update procedure is directed by β_l^k and β_g as shown in Eq. (5). Here, $s_i^k(t)$ and $s_i^k(t+1)$ are the positions of i^{th} star of k^{th} sub population at time instance t and $t+1$ respectively. r_1 and r_2 are random numbers in (0,1) range while α_l^k and α_g are the attraction forces applied on $s_i^k(t)$ by β_l^k and β_g respectively. The incorporation of local best in position update operation keeps the stars diverse by slowing down the convergence speed and sustains their exploratory behavior. As presented in Fig. 3, $s_i^k(t)$ is the position of a star in two dimensional space, $\beta_l^k(t)$ and $\beta_g(t)$ are local and global blackholes respectively at time t. The star's updated position ($s_i^k(t+1)$) obtained from both SB and SDWF are

shown through dotted (red) and dashed (blue) lines respectively where SDWF converges gradually and improves the diversity.

$$s_{(i,j)} = lb_j + r \times (ub_j - lb_j) \tag{4}$$

$$s_i^k(t+1) = s_i^k(t) + \alpha_l^k r_1 \left(\beta_l^k(t) - s_i^k(t)\right) + \alpha_g r_2 \left(\beta_g(t) - s_i^k(t)\right) \tag{5}$$

Further, the updated stars are evaluated and their MSE are computed. If k^{th} cluster finds a better solution, the respective β_l^k is replaced and β_g is updated, if applicable. In nature, any thing that enters into a blackhole is collapsed even if it is light. The blackhole optimization algorithm follows the same concept and does not allow any candidate solution to come back from an event horizon area of blackhole solution defined by its radius. The radius (r_h) of a blackhole solution is observed by computing the ratio between its fitness value and population fitness value that is the sum of all fitness values. The event horizon radius of β_l^k ($r_h\left(\beta_l^k\right)$) is computed using Eq. (6) where the fitness value of β_l^k ($f_{\beta_l^k}$) is divided by the sum of fitness values of cluster members. Similarly, the event horizon radius of global blackhole ($r_h(\beta_g)$) is computed using Eq. (7) where f_{β_g} (the fitness value of global blackhole) is divided by entire population fitness. In order to ensure whether a candidate solution has reached into the event horizon of the blackhole solution, the distance between both solutions is measured by observing the arithmetic difference of their fitness values. Since a solution is attracted towards two blackhole nodes i.e. local and global, the distance from these two blackholes must be calculated. The distance ($d_{\beta_l}(s_i^k)$) of a star s_i^k from its local blackhole (β_l^k) is computed using Eq. (8) whereas the distance ($d_{\beta_g}(s_i^k)$) from global blackhole is calculated using Eq. (9). If the distance between candidate solution s_i^k and local blackhole β_l^k denoted as $d_{\beta_l}(s_i^k)$ is less or equal to the event horizon radius of β_l^k, the s_i^k gets collapsed and a new random solution is generated to replace s_i^k to keep the number of solutions uniform throughout the simulation. Similarly, the s_i^k also gets collapsed and new random solution replaces s_i^k if it enters into the event horizon radius of the global blackhole $r_h(\beta_g)$. The operational summary of the process is mentioned in Algorithm 1.

$$r_h\left(\beta_l^k\right) = f_{\beta_l^k} / \sum_{i=1}^{P/c} f_i^k \tag{6}$$

$$r_h(\beta_g) = f_{\beta_g} / \sum_{k=1}^{c} \sum_{i=1}^{P/c} f_i^k \tag{7}$$

$$d_{\beta_l}(s_i^k) = f_{\beta_l^k} - f_{s_i^k} \tag{8}$$

$$d_{\beta_g}(s_i^k) = f_{\beta_g} - f_{s_i^k} \tag{9}$$

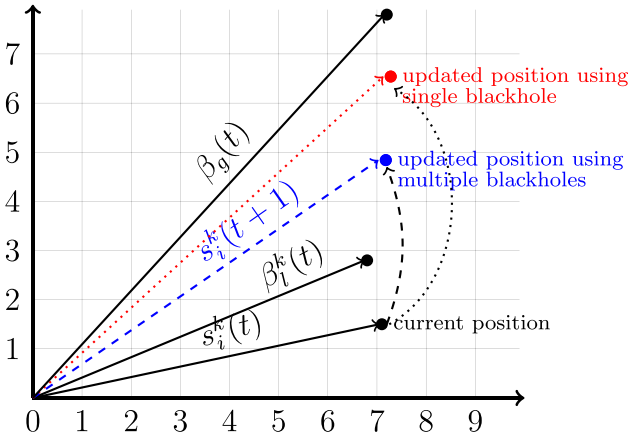


Fig. 3. Position update in SDWF predictive model.

Algorithm 1 Operational Summary of SDWF Method

1: randomly initialize S={$s_1, s_2, \ldots, s_P$}
2: organize S into k={1,2,. . .,c} clusters and evaluate each s_i^k on training data
3: **for** $k = 1, 2, \ldots, c$ **do**
4: $\beta_l^k = Best\left(s_1^k, s_2^k, \ldots, s_{P/c}^k\right)$
5: **end for**
6: $\beta_g = Best\left(\beta_l^1, \beta_l^2, \ldots, \beta_l^c\right)$
7: **while** Termination criteria is not met **do**
8: update position of each s_i^k using (5)
9: evaluate updated stars $S(t+1)$ on training data
10: **for** $k = 1, 2, \ldots, c$ **do**
11: $\beta_l^k(t+1) = Best\left(\beta_l^k(t), s_1^k(t+1), s_2^k(t+1), \ldots, s_{P/c}^k(t+1)\right)$
12: **end for**
13: $\beta_g(t+1) = Best\left(\beta_g(t), \beta_l^1(t+1), \beta_l^2(t+1), \ldots, \beta_l^c(t+1)\right)$
14: compute the radius and distances using (6)–(9)
15: **for** $i = 1, 2, \ldots, P$ **do**
16: **if** $(d_{\beta_l}(s_i^k) \leqslant \beta_l^k) \vee \left(d_{\beta_g}(s_i^k) \leqslant \beta_g\right)$ **then**
17: collapse s_i^k and regenerate using (4)
18: **end if**
19: **end for**
20: **end while**

3.3. Complexity analysis

This section, presents the time complexity of the proposed prediction approach. The first step (line 1) initializes the population of P networks each of length D and it requires $\mathcal{O}(P \times D)$. By putting $D = p(n+2)$, it becomes $\mathcal{O}(P \times p \times (n+2)) \Rightarrow \mathcal{O}(n \times p \times P)$. Since $p < n$, it can be represented as $\mathcal{O}(n^2 \times P)$. Next, line 2 organizes the population into c clusters and evaluate each solution. The evaluation of a solution includes the computing arithmetic mean of squared errors of the forecast of m samples. The computation of one forecast point takes $\approx \mathcal{O}(n^2)$ because of $p < n$ and $o < p$. Since each network evaluates m data points, the total amount of elapsed time in evaluating the solutions of one cluster would be $\mathcal{O}(m \times n^2 \times P/c)$. Therefore the total time elapsed by line 2 would be $\mathcal{O}(m \times n^2 \times P)$. Next, lines 3–5 select the local blackhole for each cluster. The selection of local blackhole for one cluster takes $\mathcal{O}(P/c)$. Therefore it takes $\mathcal{O}(P)$ to find out the local blackholes for c clusters. Further, line 6 finds the global blackhole and takes $\mathcal{O}(c)$. Line 8 updates the position of each blackhole and takes $\mathcal{O}(D)$. The next line evaluates the updated networks and takes $\mathcal{O}(m \times n^2 \times P)$. Similar to lines 3–6, lines 10–13 also consume $\mathcal{O}(P) + \mathcal{O}(c)$. Line 14 takes $\mathcal{O}(c) + \mathcal{O}(1) + \mathcal{O}(P) + \mathcal{O}(P) \Rightarrow \mathcal{O}(P)$ and lines 15–19 elapse $\mathcal{O}(P \times D)$. The lines 8–19 iterates g times, therefore the total time consumed can be computed by summing up all individual complexity and multiplying it with g. Thus, it becomes $\mathcal{O}(g \times m \times n^2 \times P)$. The total computing time complexity of the algorithm is remained to be $\mathcal{O}(g \times m \times n^2 \times P)$ after adding other individual complexities as well. We compared the complexity of the proposed workload prediction approach with neural network based forecasting models trained by Blackhole Optimization Algorithm, Differential Evolution (DE), Genetic Algorithm (GA) and Back Propagation. The complexity of the blackhole optimization algorithm is similar to the improved blackhole algorithm. The complexity of both DE and GA was founded in a similar order as $\mathcal{O}(g \times P)$ is the only additional time required in the proposed approach. We found that the time complexity of back propagation neural network (BPNN) based workload prediction approach is lesser than the proposed approach by a factor of P. The time complexity of BPNN based approach is $\mathcal{O}(g \times m \times n^2)$. But the proposed approach achieved an improved accuracy over BPNN and other methods, as discussed in the following section.

3.4. An example

Let us consider a predictive model composed with 3-2-1 network structure i.e. 3 input, 2 hidden and 1 output neurons. The length of a star or candidate solution is computed using $((n+1) \times p) + p \times q \Rightarrow ((3+1) \times 2) + 2 \times 1 \Rightarrow 10$. Considering that the population has 9 candidate solutions organized into 3 clusters (s_i^k represents i^{th} solution of cluster k).

Initial population (S)										
s_1^1	0.762	−0.469	−0.525	0.436	−0.130	−0.793	0.247	0.235	0.551	−0.819
s_2^1	0.090	0.091	−0.212	0.935	0.749	−0.288	−0.555	0.941	0.231	−0.404
s_3^1	0.572	0.378	0.443	0.677	−0.741	0.133	0.229	0.551	0.760	−0.804
s_1^2	−0.563	0.067	−0.739	0.260	0.991	0.929	0.043	−0.316	0.585	−0.621
s_2^2	−0.088	0.139	−0.589	−0.703	−0.062	0.797	−0.860	−0.921	−0.290	0.763
s_3^2	0.763	0.622	0.325	−0.748	0.040	0.942	0.917	0.486	−0.457	−0.925
s_1^3	−0.716	−0.515	−0.576	−0.103	−0.405	0.355	−0.680	−0.854	0.459	−0.293
s_2^3	0.229	0.096	0.426	−0.144	−0.669	−0.825	−0.012	−0.889	0.751	0.444
s_3^3	0.616	−0.472	0.631	0.878	0.478	−0.098	0.613	0.812	−0.468	−0.881

Each s_i^k is evaluated on a randomly generated data and mean squared error is computed to measure the forecast accuracy. Since the aim is to minimize the mean squared prediction error, the candidate solution producing the lowest forecast error is considered to be the blackhole. Therefore $s_3^1, s_2^2,$ ands_2^3 are local blackholes for their respective clusters and s_2^2 is global blackhole as it is producing the best forecasts among all local blackholes. Then positions are updated of each individual and listed as $S(t+1)$ that are again evaluated on the objective function. The fitness value of these individuals are listed as $f_{s_i^k}(t+1)$.

Fitness of initial population ($f_{s_i^k}$)								
0.0951	0.1009	0.0907	0.1173	0.0884	0.1965	0.0979	0.0943	0.1848

After fitness evaluation, the blackholes are updated and new local blackholes are $s_3^1, s_2^2,$ and s_3^3. It can be noticed that the β_l^3 is updated and found better solution. However, β_g is not changed. Next, the radius of each blackhole solution is computed and it was observed $r_h(\beta_l^1) = 0.3230, r_h(\beta_l^2) = 0.3227, r_h(\beta_l^3) = 0.3270,$ and $r_h(\beta_g) = 0.1066$. Then each individual's distance from local and global blackholes are measured. A solution except for blackholes that enter into the event horizon radius of its local or global blackhole gets collapsed, and a random solution is generated. This process is repeated till the desired solution is not achieved, or termination criteria is not met.

New population ($S(t+1)$)										
$s_1^1(t+1)$	0.079	0.171	−0.367	−0.373	−0.209	0.605	−0.593	−0.571	−0.040	0.379
$s_2^1(t+1)$	0.080	0.115	−0.252	0.601	0.513	−0.054	−0.574	0.556	0.157	−0.197
$s_3^1(t+1)$	0.145	0.223	−0.225	−0.216	−0.302	0.563	−0.476	−0.401	0.081	0.210
$s_1^2(t+1)$	−0.290	0.108	−0.653	−0.293	0.386	0.853	−0.476	−0.664	0.082	0.174
$s_2^2(t+1)$	−0.088	0.139	−0.589	−0.703	−0.062	0.797	−0.860	−0.921	−0.290	0.763
$s_3^2(t+1)$	0.107	0.250	−0.380	−0.713	−0.039	0.830	−0.454	−0.599	−0.328	0.377
$s_1^3(t+1)$	−0.258	−0.062	−0.520	−0.485	−0.205	0.559	−0.751	−0.899	0.004	0.423
$s_2^3(t+1)$	0.222	0.097	0.405	−0.156	−0.656	−0.791	−0.030	−0.890	0.729	0.451
$s_3^3(t+1)$	0.005	0.113	−0.318	−0.530	−0.191	0.391	−0.622	−0.868	−0.049	0.645

Fitness values of $S(t+1)$ ($f_{s_i^k}(t+1)$)								
0.0914	0.0973	0.0921	0.0918	0.0884	0.0938	0.0910	0.0939	0.0898

Distances									
d_{β_l}	0.0007	0.0066	0.0014	0.0033	0.0000	0.0054	0.0011	0.0040	0.0000
d_{β_g}	0.0030	0.0089	0.0037	0.0033	0.0000	0.0054	0.0026	0.0055	0.0014

4. Simulation results

This section discusses the experimental findings in detail. First of all, it describes the datasets used in this study. Further, the experimental setup is presented along with a list of key parameters and their values. Next, the forecast accuracy of the proposed scheme is evaluated over six different data sets followed by a comparative analysis with state-of-art methods including deep learning.

4.1. Dataset description

The data sets used in this study are HTTP web logs from three different World Wide Web servers (NASA's Kennedy Space Center (D_1); Department of Computer Science, University of Calgary (D_2); University of Saskatchewan (D_3)), Google cluster traces (CPU (D_4) and memory (D_5)) and PlanetLab CPU utilization trace (D_6).

NASA server traces are two HTTP request traces consisting of two months requests to the NASA Kennedy Space Center WWW server in Florida. However, only one month data is considered due to limitation of computing resource availability. Each record contains *host*, *timestamp*, *request*, *HTTP reply code*, and *bytes*. Here, *host* indicates the hostname or an IP address of the requesting machine, *timestamp* denotes the time of the request in the format "DAY MON DD HH:MM:SS YYYY" with timezone −0400, the *request* is mentioned in quotes, next field indicates the reply code, and the last field shows the bytes sent in reply of the request. Similarly, Calgary and Saskatchewan traces contain HTTP requests worth of one year and seven month's HTTP requests to the University of Calgary's department of computer science WWW server and University of Saskatchewan's WWW server respectively. Both servers are respectively located at Calgary, Alberta, Canada and Saskatoon, Saskatchewan, Canada.

The Google cluster trace was released by Google in 2011 that incorporates the data of 29 days collected from the organization's cluster cell. The data contains 10388 machines, over 0.67 million jobs, 20 million tasks. Here, a job may consist of multiple tasks, and each task may further have multiple processes and arranged in a set of six different tables.

The PlanetLab data trace contains the mean CPU utilization of more than 1000 virtual machines sampled over 5 min interval. The virtual machines are located at more than 500 places across the globe. The data was collected on 10 random days in the months of March and April of the year 2011. The experiments are carried out over randomly selected 22 virtual machines' data (see Table 1).

4.2. Experimental setup

The experiments are carried out on a machine equipped with Intel® Core™ I7-7700 processor with 3.60 GHz clock speed and 16 GB of main memory. The implementation and simulations are performed using MATLAB R2017a. The statistical analysis is also conducted using SPSS software package to validate the improvements achieved in forecast accuracy. The forecast accuracy of the proposed method is evaluated on the different Prediction Window Size (PWS) that is the time interval between two consecutive forecasts. The experiments with different number of input neurons are conducted and it is observed that network constructed with 10 input neurons produced better forecasts. The number of hidden neurons is 2/3 of input neurons as reported in [14]. The values of learning parameters are selected based on experimental analysis conducted using different combinations of learning rates. We experimented with all combinations of 0.3 and 0.8 learning rates and observed a better network performance with $\alpha_l^k = 0.3$ and $\alpha_g = 0.8$. Other parameters such as number of maximum iterations ($g = 250$), population size ($P = 100$) and number of clusters ($c = 4$) are selected randomly. The values of key parameters are listed in Table 2.

4.3. Evaluation metrics

4.3.1. Mean squared error

It is one of the well-known metrics to measure the accuracy of prediction models, which puts a high penalty on large error terms. The model is considered to be more accurate if its score is closer to 0. The mathematical representation of the metric is mentioned in Eq. (10) where m is the number of data points in workload trace.

Table 1
List of data traces and corresponding acronyms.

Name/Type of Dataset	Trace	Acronym
Web Server Traces	NASA Server	D_1
	Calgary Server	D_2
	Saskatchewan Server	D_3
Google Cluster Traces	CPU Requests	D_4
	Memory Requests	D_5
PlanetLab	CPU Utilization	D_6

Table 2
Values of key parameters.

Variable	Value
Input Neurons (n)	10
Hidden Neurons (p)	7
Population Size (ps)	100
Maximum Iterations (g)	250
Clusters (c)	4
Learning Rate (α_i^k)	0.3
Learning Rate (α_g)	0.8

$$MSE = \frac{1}{m}\sum_{t=1}^{m}(\hat{y}_t - y_t)^2 \tag{10}$$

4.3.2. Mean absolute error

In mean squared error the square of higher error values will get more weight which may influence the accuracy of forecaster due to few large errors. While MAE gives equal weight to each error component and measures the accuracy of the forecasting model by computing the mean of absolute differences between actual and predicted workloads as shown in Eq. (11). It produces a non-negative number to evaluate the forecast accuracy and if it is close to 0, forecasts are very much similar to actual values.

$$MAE = \frac{1}{m}\sum_{t=1}^{m}|y_t - \hat{y}_t| \tag{11}$$

4.3.3. Mean absolute percent error

Mean Absolute Percent Error (MAPE) computes the forecast error by dividing the absolute difference between actual and predicted values by actual values.

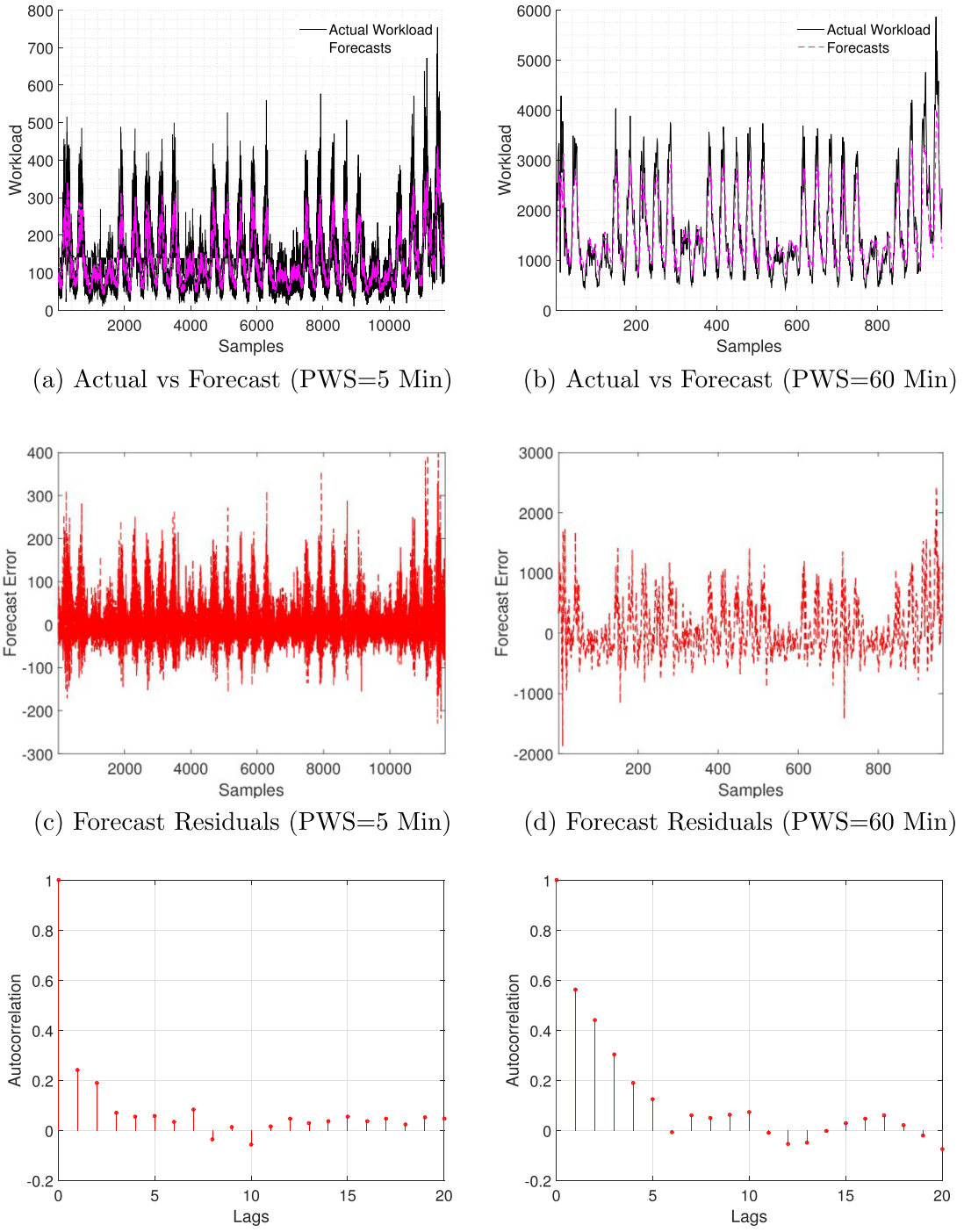
$$MAPE = \frac{100}{m}\sum_{t=1}^{m}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \tag{12}$$

4.4. Forecast evaluation

A series of experiments are carried out with the different lengths of PWS to study the performance behavior of the proposed model. The prediction window size defines the forecast time interval, e.g. if the length of the window is 10 min, the model predicts the upcoming workload on the server after every 10 min. The experiments are performed with a prediction window of 5, 10, 20, 30, 40, 50 and 60 min duration. The data trace is divided into two parts for training and testing respectively. The training data amount the 60% of total data and remaining data is used as test data.

The forecast results on NASA trace for a prediction window size of 5 and 60 min are shown in Figs. 4a and b that depict the actual and predicted workloads. The forecaster learns the trend in actual workloads and predicts a similar pattern and produces minimal residuals. The forecast errors on NASA trace for 5 and 60 min prediction intervals are depicted in Figs. 4c and d respectively. The minimum mean squared error attained by the forecasting approach is 4.12E−03 for NASA trace as shown in Table 3. The presence of autocorrelation in forecast residuals is also modeled to verify the accuracy of the model. Figs. 4e and f depict the autocorrelation in corresponding forecasts. The residual autocorrelation determines the presence of autocorrelation as of a white noise series. If it is the case, the forecasting model has successfully modeled the series and forecasts are correct. On the other hand, it is assumed that model has skipped some details in modeling the series if the residual autocorrelation is not similar to a white noise series autocorrelation. On close observation, it can be easily validated that the presence of autocorrelation is shallow in the generated residuals.

Similarly, for the Calgary data trace, the forecast results are shown in Figs. 5a and b for 5 and 60 min forecast intervals respectively. The forecast results indicate that the estimates follow a similar trend as of actual workloads that can be verified from forecast errors correspondingly depicted for 5 and 60 min interval in Figs. 5c and d. The autocorrelation is computed to analyze the forecast errors and no high correlation is observed. However, the 60 min prediction interval forecast residuals have higher autocorrelation than 5 min prediction interval forecast residuals as shown in Figs. 5e and f. The minimum forecast error achieved by the forecasting module is 2.70E−03. In the case of the Saskatchewan trace, the forecast results are rendered in Figs. 6a and b respectively for both prediction interval. The model produced the minimum mean squared error for prediction window of 1 min as listed in Table 9. The forecast residuals and their corresponding autocorrelations are shown in Figs. 6c–f and the presence of residual autocorrelation is observed in 60 min interval forecast which indicates that the prediction model can be improved or training data should be increased.



(a) Actual vs Forecast (PWS=5 Min) (b) Actual vs Forecast (PWS=60 Min)

(c) Forecast Residuals (PWS=5 Min) (d) Forecast Residuals (PWS=60 Min)

(e) Residuals autocorrelation (PWS=5 Min) (f) Residuals autocorrelation (PWS=60 Min)

Fig. 4. Prediction results for NASA trace.

Similar to web servers' HTTP request workloads, a set of experiments is also conducted for CPU and memory request traces of Google cluster and CPU utilization of PlanetLab data trace. Fig. 7 shows the forecast results of CPU request trace of Google cluster. The model captures the trend of workload on the servers. The forecast residuals and their autocorrelation indicate the presence of autocorrelation. Similarly, memory request trace results are depicted in Fig. 8 and similar observations are found. It is noticed that the 5 min interval forecasts are having more high error terms than 60 min interval forecasts for both data traces. The minimum mean squared error achieved by the model is 5.14E−04 and 2.19E−04 for CPU and mem-

Table 3
Forecast accuracy of the proposed model measured using MSE.

PWS (min)	D_1	D_2	D_3	D_4	D_5	D_6
1	1.09E−02	2.70E−03	9.04E−04	5.14E−04	2.19E−04	NA
5	4.12E−03	4.60E−03	1.42E−03	1.19E−03	2.40E−03	1.04E−02
10	4.24E−03	5.96E−03	2.41E−03	1.80E−03	3.41E−03	1.21E−02
20	4.71E−03	7.38E−03	2.46E−03	3.84E−03	6.91E−03	1.31E−02
30	5.06E−03	1.03E−02	2.47E−03	5.45E−03	7.08E−03	1.43E−02
40	6.37E−03	9.61E−03	2.75E−03	8.42E−03	8.76E−03	1.46E−02
50	9.03E−03	8.57E−03	3.79E−03	8.48E−03	1.17E−02	1.64E−02
60	8.57E−03	1.13E−02	4.55E−03	1.07E−02	1.08E−02	1.64E−02

ory data traces. Fig. 9 shows the forecast results on CPU utilization of the PlanetLab data trace. It can be seen that only 60 min interval forecasts have significant residual autocorrelation. Similarly, the forecast results for D_6 are depicted in Fig. 9 and it can be easily determined that the forecaster learns the trend of actual mean CPU utilization over time. The forecasts are close to the actual CPU utilization in most of the instances but residuals are correlated. However, the model produced forecasts with significant residuals for 60 min interval. Moreover, the model is also evaluated on Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE). The MAE ignores the direction of forecast error while measuring the average of errors. Table 4 reports the average of absolute errors observed in test samples of the data and it can be seen that theroposed model achieves low forecast errors. On the other hand, MAPE uses MAE and reports the forecast errors in percentage. Table 5 shows the MAPE observed in the forecasts of the proposed approach. The results verify that the model forecasts the upcoming workload with low forecast errors. Besides, the amount of time elapsed in training is also computed that is listed in Table 6. The training time can be easily reduced by adopting the high computing infrastructure and parallel implementation of the approach.

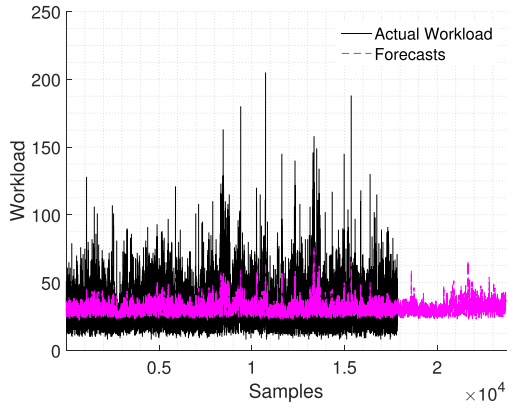
4.5. Comparative analysis

In order to evaluate the efficacy of the proposed method, its performance is compared with prediction methods based on following state-of-art learning algorithms.

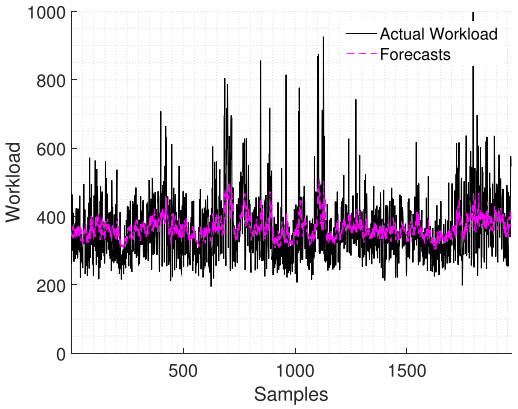
- Blackhole Algorithm [18]: The optimization approach is inspired by the blackhole phenomena in nature. The proposed approach improves the standard version of the blackhole algorithm to learn the network weights more effectively. The comparative analysis is carried out with the forecasting results obtained using the standard version of blackhole algorithm.
- Deep Learning [34,31]: A deep learning architecture is usually a special kind of neural network composed of multiple layers. The deep learning based forecasting approach is used to compare the forecast accuracy of the proposed method on PlanetLab CPU utilization trace. In this study, the performance of the proposed approach is also compared on D_1–D_5 with long short term memory recurrent neural network (LSTM-RNN) based forecasting method.
- Differential Evolution [14]: It a population based algorithm developed by Storn and Price. In this method, the solutions are encoded into vector form that explores the search space by combining the positions of different solutions from the population. The performance of the proposed approach is compared with self adaptive differential evolution (SaDE) based forecasting method.
- Back-Propagation Algorithm [17]: It computes the error gradient and feeds it back to adjust the network weights. It has widely used in the training of neural networks and deep neural networks. The performance of the proposed approach is also compared with backpropagation based predictive model.

$$E_r = \frac{E_{st} - E_{pr}}{E_{st}} * 100 \tag{13}$$

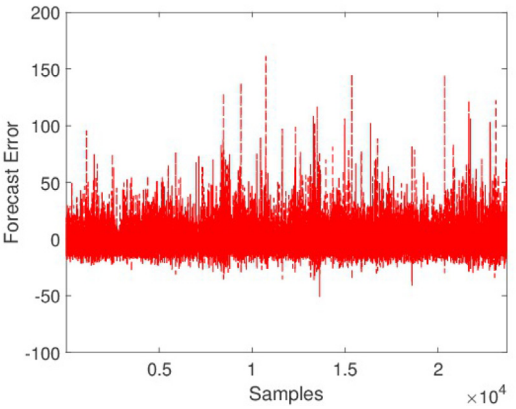
The forecast accuracy of the proposed model with other state-of-art approaches compared and corresponding results are shown in Tables 7–12 for each data trace respectively. The results indicate that the proposed model improves the forecast accuracy in most of the experiments. The ratio of mean squared error reduction is computed using Eq. (13), where E_{pr} and E_{st} denote the mean squared error obtained by the proposed model and a state-of-art model. The maximum relative reduction attained by the proposed approach for NASA trace is $36.34\%, 64.67\%, 83.04\%$ and 98.64% over deep learning, Blackhole, SaDE and BPNN approaches respectively. Similarly $21.05\%, 59.73\%, 9.45\%$ and 99.09% for Calgary trace, $81.92\%, 72.11\%$, 94.35% and 99.58% for Saskatchewan trace over LSTM, Blackhole, SaDE and BPNN approaches respectively. For Google cluster trace, the maximum reduction in mean squared error is noticed up to $51.77\%, 39.39\%$ and 89.61% for CPU trace while $60.16\%, 25.55\%$ and 88.88% for Memory trace over Blackhole, SaDE and BPNN models. Also, a notable reduction in forecast error is observed on comparing the performance of the proposed approach with deep learning based forecasting method on mean CPU utilization of PlanetLab trace. The proposed approach reduced the forecast error by up to 99.99%.



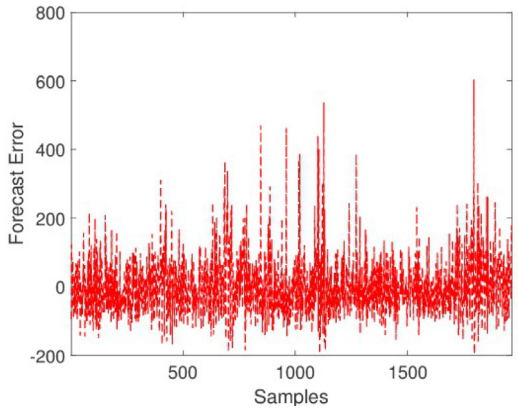
(a) Actual vs Forecast (PWS=5 Min)



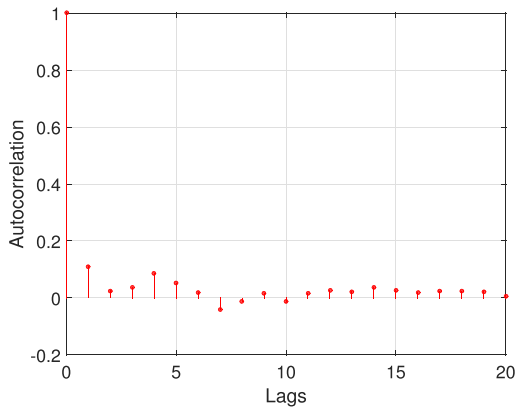
(b) Actual vs Forecast (PWS=60 Min)



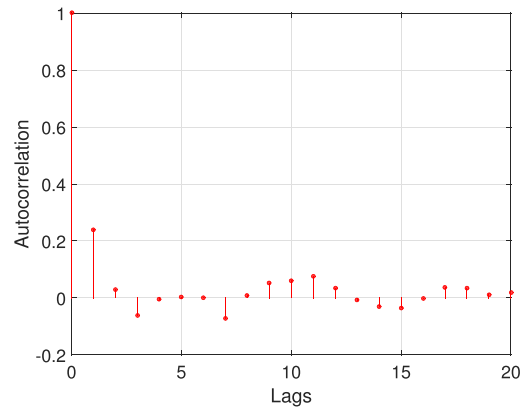
(c) Forecast Residuals (PWS=5 Min)



(d) Forecast Residuals (PWS=60 Min)



(e) Residuals autocorrelation (PWS=5 Min)

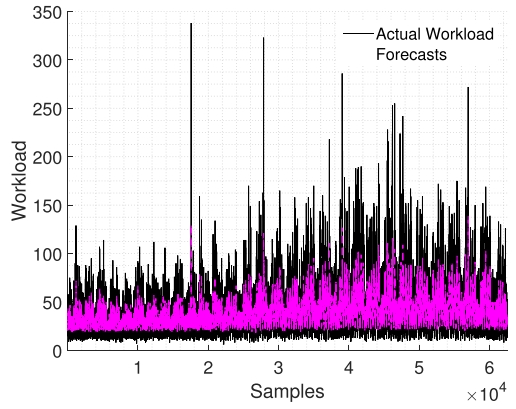


(f) Residuals autocorrelation (PWS=60 Min)

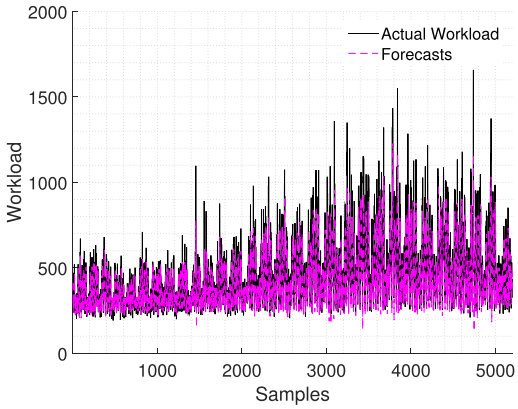
Fig. 5. Prediction results for Calgary trace.

5. Statistical analysis

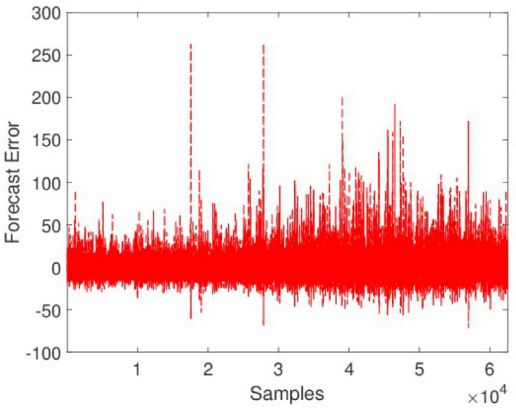
The Friedman test [48] is used to study the statistical behavior of the predictive models that conducts a multiple comparison analysis to detect the presence of a significant difference in the performance of the multiple algorithms. The method



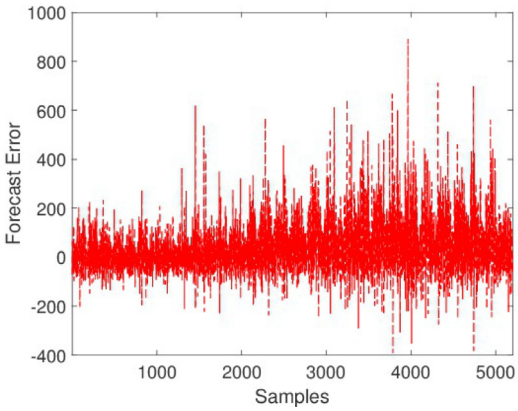
(a) Actual vs Forecast (PWS=5 Min)



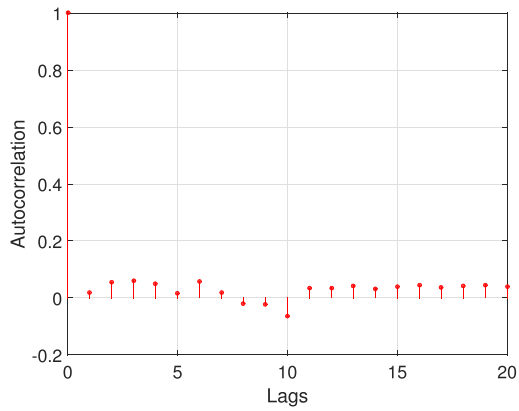
(b) Actual vs Forecast (PWS=60 Min)



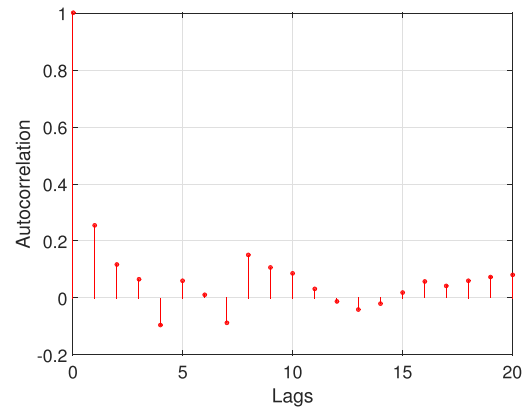
(c) Forecast Residuals (PWS=5 Min)



(d) Forecast Residuals (PWS=60 Min)



(e) Residuals autocorrelation (PWS=5 Min)

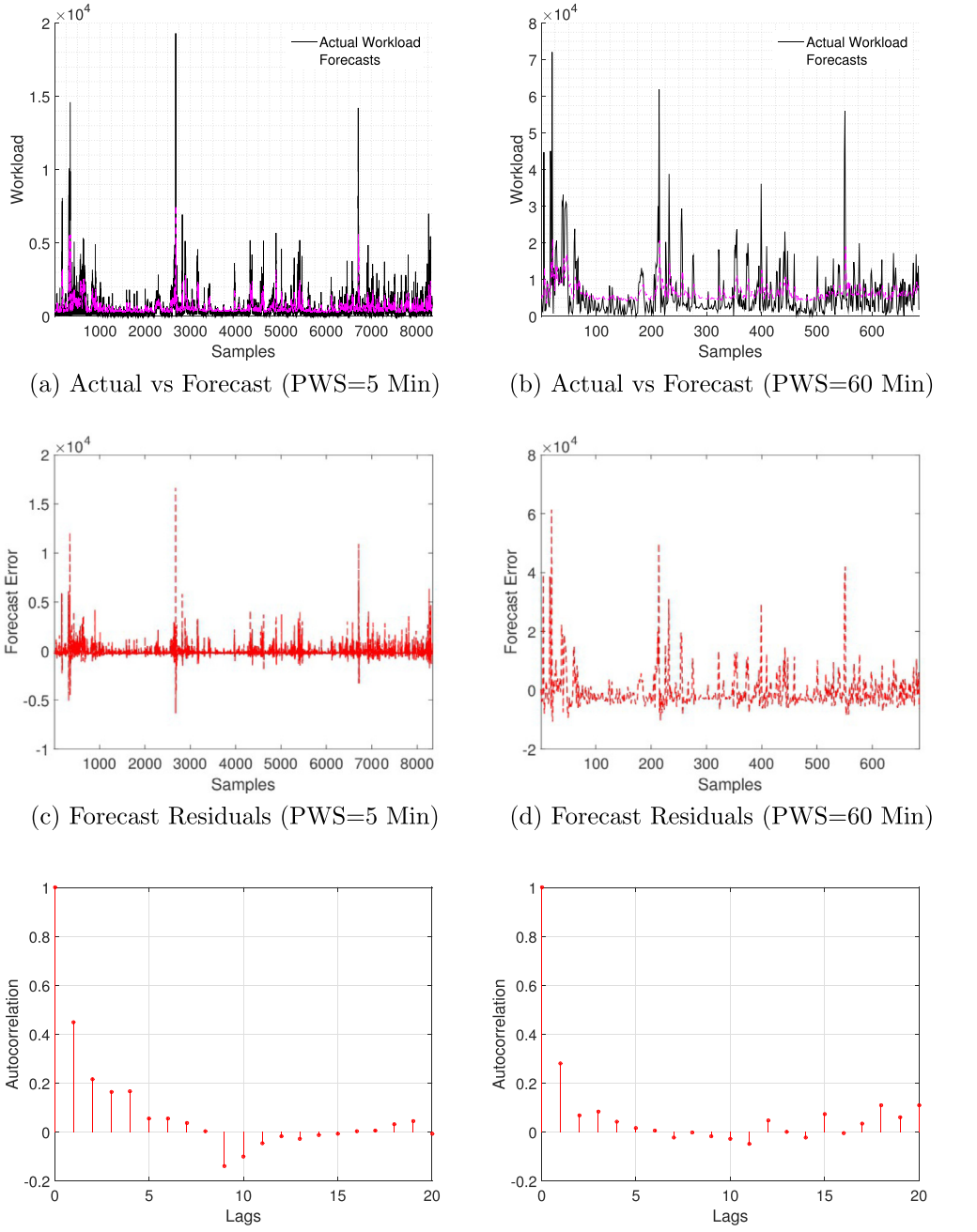


(f) Residuals autocorrelation (PWS=60 Min)

Fig. 6. Prediction results for Saskatchewan trace.

assumes a null hypothesis (H_0) which states that the mean of all algorithms' results are equal and an alternate hypothesis (H_1) that negates the H_0.

First, the Friedman mean rank of each algorithm is computed that is listed in Table 13. It can be seen from the table that the proposed approach achieved better rank over LSTM, Blackhole, SaDE and BPNN based predictive models except for D_2 where LSTM based model produced better forecasts than the proposed approach. Along with the ranks, the Friedman test



(a) Actual vs Forecast (PWS=5 Min) (b) Actual vs Forecast (PWS=60 Min)

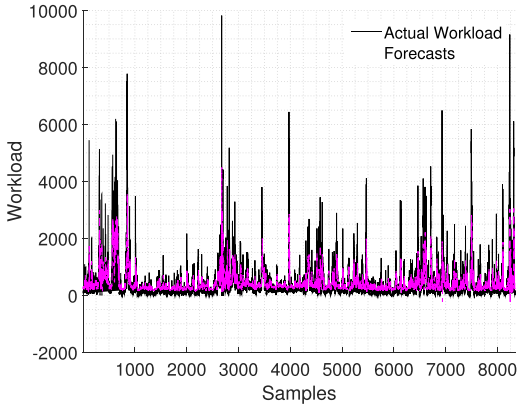
(c) Forecast Residuals (PWS=5 Min) (d) Forecast Residuals (PWS=60 Min)

(e) Residuals autocorrelation (PWS=5 Min) (f) Residuals autocorrelation (PWS=60 Min)

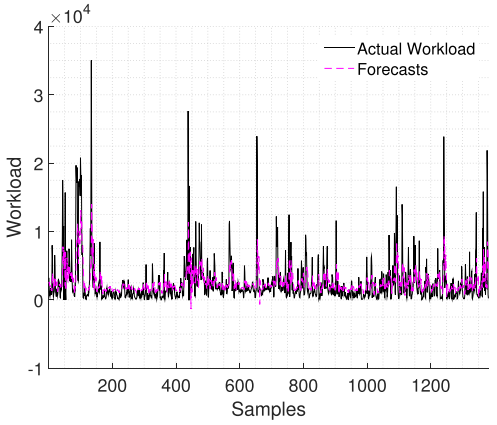
Fig. 7. Prediction results for CPU request trace of Google cluster.

statistics (χ^2 and p-value) are mentioned in Table 14. The p-values indicated that the H_0 is rejected with significance level ($\alpha = 0.05$) and the ranks clearly indicate that the behavior of the proposed approach has significant difference.

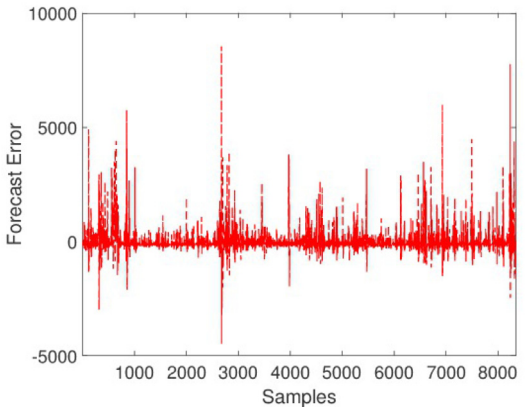
As it has been already observed that the results of the proposed method are significantly different, the Wilcoxon signed test [49] is conducted to ensure that the difference is positive. The null hypothesis (H_0) of the test is that the mean of the results of the paired approaches are same. The test conducts the pairwise test and computes the positive (R^+) and negative (R^-) mean rankings. These rankings are calculated using the difference in the results of the two algorithms. Let M_p and M_c are two approaches (proposed and competitor), M_p wins only if the difference between M_p and M_c is positive. The rankings (Table 15) are computed based on these differences, where competitors are listed in the first column. The corresponding R^+, R^- and p-values are shown in Table 15. As the table states, the proposed approach produces significant differences over



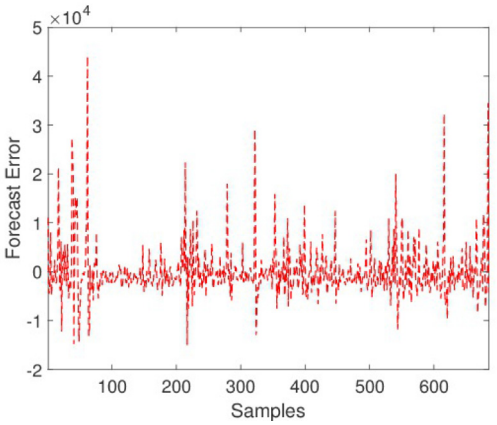
(a) Actual vs Forecast (PWS=5 Min)



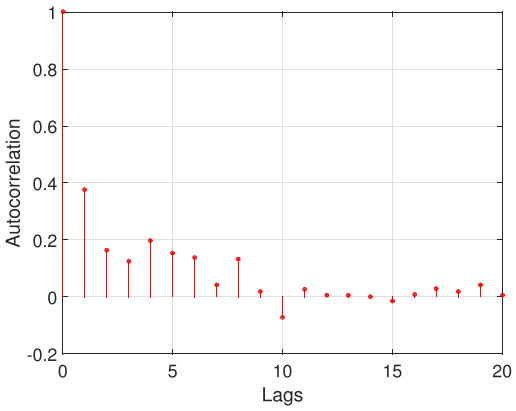
(b) Actual vs Forecast (PWS=60 Min)



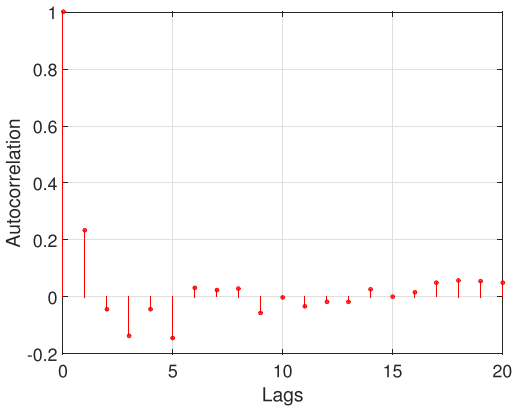
(c) Forecast Residuals (PWS=5 Min)



(d) Forecast Residuals (PWS=60 Min)



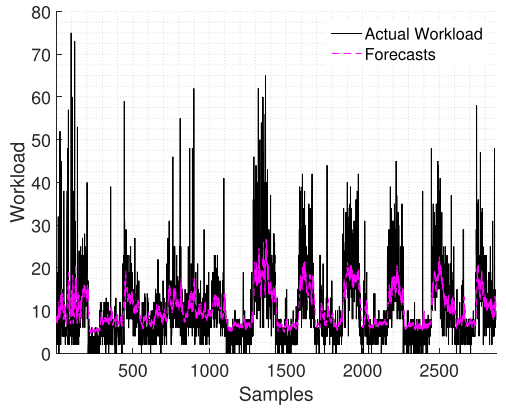
(e) Residuals autocorrelation (PWS=5 Min)



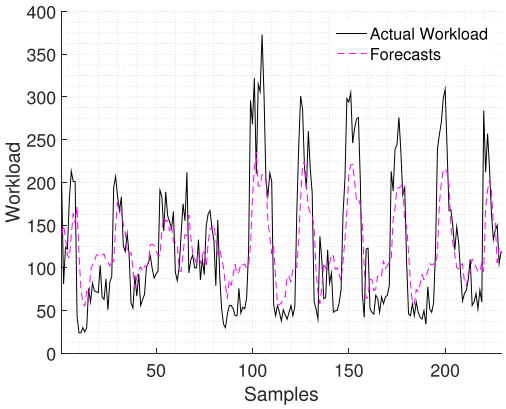
(f) Residuals autocorrelation (PWS=60 Min)

Fig. 8. Prediction results for Memory request trace of Google cluster.

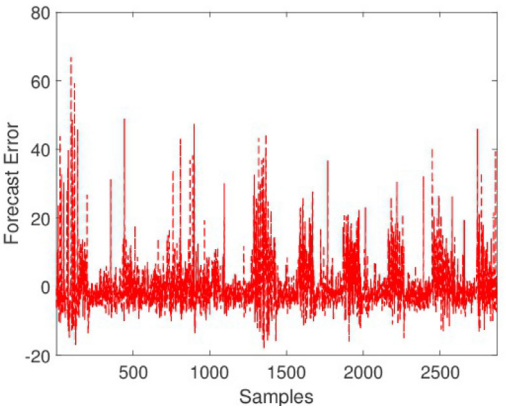
Blackhole, BPNN and SaDE with the significance level (α) 0.05, 0.05 and 0.1 respectively and rejects the H_0. While the H_0 is accepted for LSTM. On comparing the performance of the proposed approach with deep learning based method on D_6, it is observed that the H_0 is rejected with $\alpha = 0.1$. The proposed algorithm's superiority can be validated based on the promising results and their statistical analysis.



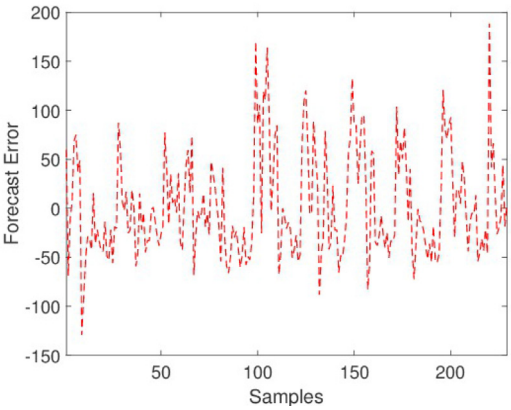
(a) Actual vs Forecast (PWS=5 Min)



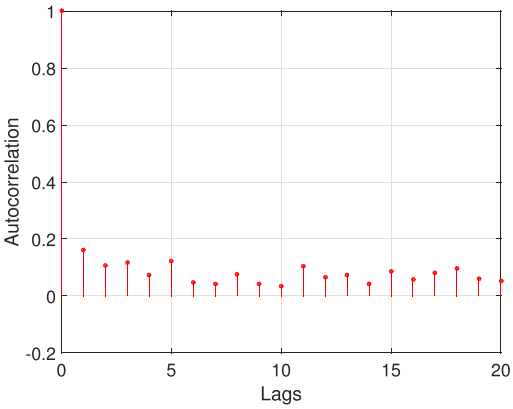
(b) Actual vs Forecast (PWS=60 Min)



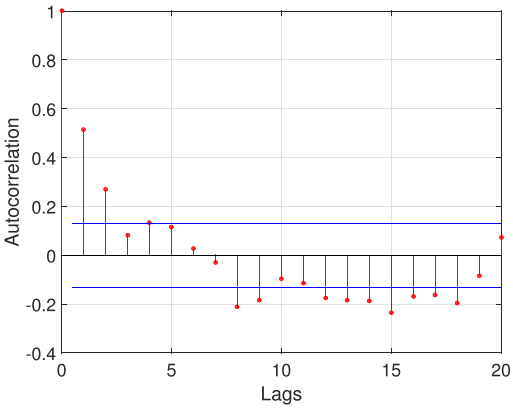
(c) Forecast Residuals (PWS=5 Min)



(d) Forecast Residuals (PWS=60 Min)



(e) Residuals autocorrelation (PWS=5 Min)



(f) Residuals autocorrelation (PWS=60 Min)

Fig. 9. Prediction results for CPU utilization of PlanetLab trace.

Table 4
Forecast accuracy of the proposed model measured using MAE.

PWS (min)	D_1	D_2	D_3	D_4	D_5	D_6
1	7.40E−02	3.59E−02	2.12E−02	1.09E−02	7.43E−03	NA
5	4.56E−02	4.90E−02	2.72E−02	1.52E−02	2.73E−02	8.10E−02
10	4.85E−02	5.45E−02	3.60E−02	1.66E−02	3.04E−02	8.70E−02
20	4.68E−02	5.92E−02	3.48E−02	2.79E−02	4.58E−02	9.90E−02
30	5.52E−02	6.85E−02	3.46E−02	3.10E−02	4.18E−02	1.04E−01
40	5.85E−02	6.38E−02	3.85E−02	4.54E−02	5.47E−02	1.07E−01
50	6.69E−02	6.02E−02	4.19E−02	3.95E−02	6.68E−02	1.14E−01
60	5.98E−02	6.91E−02	4.65E−02	4.78E−02	6.16E−02	1.16E−01

Table 5
Forecast accuracy of the proposed model measured using MAPE.

PWS (min)	D_1	D_2	D_3	D_4	D_5	D_6
1	1.00E−02	2.03E−04	4.40E−05	4.20E−05	9.70E−04	NA
5	1.83E−02	3.39E−03	3.63E−04	3.25E−03	1.30E−03	5.78E−02
10	2.28E−03	7.93E−03	3.04E−03	1.59E−02	6.80E−03	1.19E−01
20	5.27E−03	3.28E−03	1.43E−03	7.45E−03	4.06E−02	3.15E−01
30	2.09E−02	1.02E−02	9.86E−04	2.30E−02	4.54E−01	5.10E−01
40	2.98E−02	4.30E−04	8.68E−04	1.32E−01	8.36E−01	6.44E−01
50	4.00E−02	9.63E−03	4.12E−04	2.76E−01	2.27E−01	9.63E−01
60	5.85E−02	3.03E−02	6.79E−03	2.58E−01	2.07E−01	1.54E+00

Table 6
Time elapsed in training of the proposed model (min).

PWS (min)	D_1	D_2	D_3	D_4	D_5	D_6
1	81.68	166.43	439.7	60.61	60.3	NA
5	16.21	33.3	91.4	11.86	12.1	4.11
10	8.1	16.45	45.88	6.03	5.87	2.1
20	4.08	8.45	22.2	2.99	2.96	1.05
30	2.87	5.51	14.92	1.99	1.97	0.68
40	2.08	4.11	11.26	1.48	1.49	0.52
50	1.63	3.3	8.99	1.19	1.19	0.41
60	1.35	2.74	7.35	0.98	1	0.33

Table 7
Accuracy comparison on NASA Trace predictions.

PWS (min)	Proposed	LSTM	Blackhole	SaDE	BPNN
1	1.09E−02	1.31E−02	2.14E−02	1.69E−04	2.43E−01
5	4.12E−03	4.79E−03	8.45E−03	1.00E−02	3.02E−01
10	4.24E−03	6.66E−03	1.20E−02	2.50E−02	2.82E−01
20	4.71E−03	7.01E−03	7.78E−03	2.50E−02	3.38E−01
30	5.06E−03	6.43E−03	9.06E−03	2.02E−02	2.78E−01
60	8.57E−03	5.59E−03	2.31E−02	2.02E−02	3.34E−01

Table 8
Accuracy comparison on Calgary Trace predictions.

PWS (min)	Proposed	LSTM	Blackhole	SaDE	BPNN
1	2.70E−03	3.42E−03	6.03E−03	2.79E−03	2.97E−01
5	4.60E−03	4.10E−03	5.99E−03	5.08E−03	2.90E−01
10	5.96E−03	6.11E−03	1.48E−02	6.06E−03	2.87E−01
20	7.38E−03	5.99E−03	1.25E−02	7.79E−03	2.78E−01
30	1.03E−02	7.12E−03	1.78E−02	1.02E−02	5.00E−01
60	1.13E−02	8.03E−03	1.97E−02	1.17E−02	2.97E−01

Table 9
Accuracy comparison on Saskatchewan Trace predictions.

PWS (min)	Proposed	LSTM	Blackhole	SaDE	BPNN
1	9.04E−04	5.00E−03	1.61E−03	1.00E−06	4.02E−02
5	1.42E−03	3.17E−03	2.51E−03	4.00E−06	3.37E−01
10	2.41E−03	5.26E−03	5.55E−03	2.50E−02	2.90E−01
20	2.46E−03	5.56E−03	8.82E−03	3.84E−02	3.18E−01
30	2.47E−03	4.79E−03	8.03E−03	4.37E−02	5.07E−01
60	4.55E−03	4.50E−03	9.30E−03	2.89E−02	2.86E−01

Table 10
Accuracy comparison on CPU Trace predictions.

PWS (min)	Proposed	Blackhole	SaDE	BPNN
1	5.14E−04	9.26E−04	8.48E−04	4.41E−03
5	1.19E−03	2.19E−03	1.68E−03	8.75E−03
10	1.80E−03	3.59E−03	2.47E−03	1.42E−02
20	3.84E−03	6.28E−03	4.02E−03	2.60E−02
30	5.45E−03	1.13E−02	5.49E−03	4.37E−02
60	1.07E−02	1.43E−02	1.03E−02	1.03E−01

Table 11
Accuracy comparison on Memory Trace predictions.

PWS (min)	Proposed	Blackhole	SaDE	BPNN
1	2.19E−04	2.48E−04	2.35E−04	1.97E−03
5	2.40E−03	4.91E−03	2.23E−03	1.91E−02
10	3.41E−03	8.56E−03	4.08E−03	2.48E−02
20	6.91E−03	1.36E−02	5.54E−03	4.80E−02
30	7.08E−03	1.15E−02	9.51E−03	5.16E−02
60	1.08E−02	1.09E−02	1.18E−02	8.77E−02

Table 12
Accuracy comparison on PlanetLab Trace predictions.

PWS (min)	Proposed	Deep Learning
5	1.04E−02	8.41E+01
15	1.33E−02	9.29E+01
30	1.43E−02	1.06E+02
60	1.64E−02	9.94E+01

Table 13
Friedman mean ranks.

Prediction Model	D_1	D_2	D_3	D_4	D_5
Proposed	**1.33**	1.83	**1.50**	**1.17**	**1.33**
LSTM	2.00	**1.67**	2.50	–	–
Blackhole	3.33	4.00	3.00	3.00	2.83
SaDE	3.33	2.50	3.00	1.83	1.83
BPNN	5.00	5.00	5.00	4.00	4.00

The bold values indicate the best ranks achieved among all predictive models over respective data traces.

Table 14
Friedman statistics.

	D_1	D_2	D_3	D_4	D_5
χ^2	19.200	20.133	15.600	17.000	15.000
p	0.001	0.000	0.004	0.001	0.002

Table 15
Wilcoxon signed ranks.

	Proposed	D_1	D_2	D_3	D_4	D_5	D_6
LSTM	R^-	6.00	17.00	1.00	–	–	–
	R^+	15.00	4.00	20.00	–	–	–
	p-value	0.345	0.173	0.046	–	–	–
Blackhole	R^-	0.00	0.00	0.00	0.00	0.00	–
	R^+	21.00	21.00	21.00	21.00	21.00	–
	p-value	0.028	0.028	0.028	0.028	0.028	–
SaDE	R^-	2.00	2.50	3.00	4.00	7.00	–
	R^+	19.00	18.50	18.00	17.00	14.00	–
	p-value	0.075	0.093	0.116	0.173	0.463	–
BPNN	R^-	0.00	0.00	0.00	0.00	0.00	–
	R^+	21.00	21.00	21.00	21.00	21.00	–
	p-value	0.028	0.028	0.028	0.028	0.028	–
Deep Learning	R^-	–	–	–	–	–	0.00
	R^+	–	–	–	–	–	10.00
	p-value	–	–	–	–	–	0.068

6. Conclusions

The workload prediction has been an essential aid for efficient resource management. This paper presented a self directed workload prediction model for the cloud data centers. The model used a nature inspired heuristic approach for the purpose of training. It also proposed a solution to overcome the identified limitation of the learning algorithm. The method is adaptable that determines the error in its previous forecasts and modifies its future forecast accordingly. The forecast accuracy of the proposed model is analyzed on different time interval forecasts over multiple real world data traces. It also presents a statistical analysis to validate the performance of the proposed method. The experimental results show that the model is capable of reducing the mean squared forecasting errors up to 99.99% over existing models that validates the superiority of SDWF approach over popular state-of-art techniques. The approach may be extended further by proposing a solution to optimize the forecast error feedback window, cluster size, and learning rate. Also, other learning schemes including quantum inspired algorithms can be explored for better quality solutions.

CRediT authorship contribution statement

Jitendra Kumar: Conceptualization, Methodology, Software, Investigation, Formal analysis, Writing - original draft. **Ashutosh Kumar Singh:** Conceptualization, Methodology, Writing - review & editing, Supervision, Funding acquisition. **Rajkumar Buyya:** Conceptualization, Methodology, Writing - review & editing, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is financially supported by the Ministry of Electronics & Information Technology (MeitY), Government of India.

References

[1] A.K. Singh, J. Kumar, Secure and energy aware load balancing framework for cloud data centre networks, Electronics Letters 55 (1) (2019) 540–541.
[2] R. Birke, L.Y. Chen, E. Smirni, R. Birke, L.Y. Chen, E. Smirni, Data Centers in the Wild: A Large Performance Study, Tech. rep., IBM Research – Zurich, Switzerland (2012)..
[3] C. Reiss, A. Tumanov, G.R. Ganger, R.H. Katz, M.A. Kozuch, Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis, in: ACM Symposium on Cloud Computing 2012, 2012, pp. 1–18..
[4] L. Barroso, U. Hölzle, The Datacenter as a Computer An Introduction to the Design of Warehouse-Scale Machines, vol. 24, Morgan & Claypool Publishers, 2013.
[5] Navigant Consulting Inc. SAIC, Analysis and Representation of Miscellaneous Electric Loads in NEMS, prepared for the U.S. Energy Information Administration (Navigant Reference: 160750) (2017) 1–138..
[6] R. Buyya, J. Broberg, A.M. Goscinski, Cloud Computing Principles and Paradigms, Wiley Publishing, 2011.
[7] D.F. Kirchoff, M. Xavier, J. Mastella, C.A. F De Rose, A preliminary study of machine learning workload prediction techniques for cloud applications, in: 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2019, pp. 222–227..
[8] J. Kumar, A.K. Singh, Cloud datacenter workload estimation using error preventive time series forecasting models, Cluster Computing 23 (2020) 1363–1379.
[9] J. Bi, H. Yuan, L. Zhang, J. Zhang, SGW-SCN: An integrated machine learning approach for workload forecasting in geo-distributed cloud data centers, Information Sciences 481 (2019) 57–68.

[10] J. Kumar, A.K. Singh, Cloud resource demand prediction using differential evolution based learning, in: 2019 7th International Conference on Smart Computing Communications (ICSCC), 2019, pp. 1–5.
[11] M. Amiri, L. Mohammad-Khanli, Survey on prediction models of applications for resources provisioning in cloud, Journal of Network and Computer Applications 82 (2017) 93–113.
[12] S. Kumaraswamy, M.K. Nair, Intelligent VMs prediction in cloud computing environment, 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon) (2017) 288–294..
[13] K. Mason, M. Duggan, E. Barrett, J. Duggan, E. Howley, Predicting host CPU utilization in the cloud using evolutionary neural networks, Future Generation Computer Systems 86 (2018) 162–173.
[14] J. Kumar, A.K. Singh, Workload prediction in cloud using artificial neural network and adaptive differential evolution, Future Generation Computer Systems 81 (2018) 41–52.
[15] S. Islam, J. Keung, K. Lee, A. Liu, Empirical prediction models for adaptive resource provisioning in the cloud, Future Generation Computer Systems 28 (1) (2012) 155–162.
[16] J. Kumar, A.K. Singh, R. Buyya, Ensemble learning based predictive framework for virtual machine resource request prediction, Neurocomputing 397 (2020) 20–30.
[17] 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, 2011, pp. 276–281.
[18] J. Kumar, A.K. Singh, Dynamic resource scaling in cloud using neural network and black hole algorithm, in: 2016 Fifth International Conference on Eco-friendly Computing and Communication Systems (ICECCS), 2016, pp. 63–67..
[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 Computing and Applications 27 (8) (2016) 2383–2406.
[20] M. Dabbagh, B. Hamdaoui, M. Guizani, A. Rayes, Energy-Efficient Resource Allocation and Provisioning Framework for Cloud Data Centers, IEEE Transactions on Network and Service Management 12 (3) (2015) 377–391.
[21] H. Shen, L. Chen, Resource Demand Misalignment: An Important Factor to Consider for Reducing Resource Over-Provisioning in Cloud Datacenters, IEEE/ACM Transactions on Networking (2018) 1–15.
[22] F.-H. Tseng, X. Wang, L.-D. Chou, H.-C. Chao, V.C.M. Leung, Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm, IEEE Systems Journal 12 (2) (2018) 1688–1699.
[23] C. Liu, J. Han, Y. Shang, C. Liu, B. Cheng, J. Chen, Predicting of job failure in compute cloud based on online extreme learning machine: a comparative study, IEEE Access 5 (2017) 9359–9368.
[24] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, Proactive workload management in hybrid cloud computing, IEEE Transactions on Network and Service Management 11 (1) (2014) 90–100.
[25] J. Yang, C. Liu, Y. Shang, B. Cheng, Z. Mao, C. Liu, L. Niu, J. Chen, A cost-aware auto-scaling approach using the workload prediction in service clouds, Information Systems Frontiers 16 (1) (2014) 7–18.
[26] M. Pulido, P. Melin, O. Castillo, Particle swarm optimization of ensemble neural networks with fuzzy aggregation for time series prediction of the mexican stock exchange, Information Sciences 280 (2014) 188–204.
[27] J. Soto, P. Melin, O. Castillo, Time series prediction using ensembles of ANFIS models with genetic optimization of interval type-2 and type-1 fuzzy integrators, International Journal of Hybrid Intelligent Systems 11 (3) (2014) 211–226.
[28] C. Gupta, A. Jain, D.K. Tayal, O. Castillo, ClusFuDE: Forecasting low dimensional numerical data using an improved method based on automatic clustering, fuzzy relationships and differential evolution, Engineering Applications of Artificial Intelligence 71 (2018) 175–189.
[29] N. Liu, Z. Li, J. Xu, Z. Xu, S. Lin, Q. Qiu, J. Tang, Y. Wang, A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning, in: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2017, pp. 372–382.
[30] F. Qiu, B. Zhang, J. Guo, A deep learning approach for VM workload prediction in the cloud, in: 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), IEEE, 2016, pp. 319–324..
[31] J. Kumar, R. Goomer, A.K. Singh, Long short term memory recurrent neural network (LSTM-RNN) based workload forecasting model for cloud datacenters, Procedia Computer Science 125 (2018) 676–682.
[32] Y. Zhang, J. Yao, H. Guan, Intelligent cloud resource management with deep reinforcement learning, IEEE Cloud Computing 4 (6) (2017) 60–69.
[33] Y.S. Patel, R. Misra, Performance comparison of deep VM workload prediction approaches for cloud, in: P. Pattnaik, S. Rautaray, H. Das, J. Nayak (Eds.), Progress in Computing, Analytics and Networking. Advances in Intelligent Systems and Computing, Springer, Singapore, 2018, pp. 149–160.
[34] Q. Zhang, L.T. Yang, Z. Yan, Z. Chen, P. Li, An efficient deep learning model to predict cloud workload for industry informatics, IEEE Transactions on Industrial Informatics (2018) 1–9.
[35] Z. Xiao, S. Member, W. Song, Q. Chen, Dynamic resource allocation using virtual machines for cloud computing environment, IEEE Transactions on Parallel and Distributed Systems 24 (6) (2013) 1107–1117.
[36] S. Kim, T. Kim, C. Yoo, Workload prediction using run-length encoding for runtime processor power management, Electronics Letters 51 (22) (2015) 1759–1761.
[37] M. Amiri, L. Mohammad-Khanli, R. Mirandola, A sequential pattern mining model for application workload prediction in cloud environment, Journal of Network and Computer Applications 105 (2018) 21–62.
[38] M. Amiri, L. Mohammad-Khanli, R. Mirandola, An online learning model based on episode mining for workload prediction in cloud, Future Generation Computer Systems 87 (2018) 83–101.
[39] G. Kaur, A. Bala, I. Chana, An intelligent regressive ensemble approach for predicting resource usage in cloud computing, Journal of Parallel and Distributed Computing 123 (2019) 1–12.
[40] J. Bi, H. Yuan, M. Zhou, Temporal prediction of multiapplication consolidated workloads in distributed clouds, IEEE Transactions on Automation Science and Engineering (2019) 1–11.
[41] W. Lin, G. Wu, X. Wang, K. Li, An artificial neural network approach to power consumption model construction for servers in cloud data centers, IEEE Transactions on Sustainable Computing (2019), 1–1.
[42] X. Tang, Large-scale computing systems workload prediction using parallel improved LSTM neural network, IEEE Access 7 (2019) 40525–40533.
[43] Kumar J., Saxena D., Singh A.K., Mohan A., Biphase adaptive learning based neural network model for cloud workload forecasting, Soft Computing (2020). https://doi.org/10.1007/s00500-020-04808-9.
[44] I.K. Kim, W. Wang, Y. Qi, M. Humphrey, CloudInsight: Utilizing a Council of Experts to Predict Future Cloud Application Workloads, in: In 10th IEEE International Conference on Cloud Computing (Cloud 2018), July 2 – July 7, San Francisco, USA, 2018, pp. 1–8..
[45] W. Zhong, Y. Zhuang, J. Sun, J. Gu, A load prediction model for cloud computing using PSO-based weighted wavelet support vector machine, Applied Intelligence (2018) 1–12.
[46] A. Hatamlou, Black hole: A new heuristic optimization approach for data clustering, Information Sciences 222 (Supplement C) (2013) 175–184, including Special Section on New Trends in Ambient Intelligence and Bio-inspired Systems..
[47] X.-S. Yang, Nature-Inspired Optimization Algorithms, Elsevier, 2011.
[48] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, The Annals of Mathematical Statistics 11 (1) (1940) 86–92.
[49] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bulletin 1 (6) (1945) 80–83.

Dr. Jitendra Kumar is an Assistant Professor in the Department of Computer Applications, National Institute of Technology Tiruchirappalli, India. He earned his doctorate from the National Institute of Technology Kurukshetra, India (An Institution of National Importance) in 2019. He has published several research articles in national and international journals and conferences of high repute. His current research interests include Cloud Computing, Machine Learning, Data Analytics, Parallel Processing.

Prof. 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 18 years of research and teaching experience in various Universities of India, the UK, and Malaysia. He received his Ph.D. in Electronics Engineering from Indian Institute of Technology, BHU, India, and Post Doc from the Department of Computer Science, University of Bristol, UK. He is also a Charted Engineer from the UK. His research area includes Verification, Synthesis, Design, and Testing of Digital Circuits, Data Science, Cloud Computing, Machine Learning, Security, Big Data. He has published more than 160 research papers in different journals, conferences, and news magazines. He is the co-author of six books, which include 'Web Spam Detection Application using Neural Network,' 'Digital Systems Fundamentals', and 'Computer System Organization & Architecture.' He has worked as an Editorial Board Member of International Journal of Networks and Mobile Technologies, International Journal of Digital Content Technology and its Applications. Also, he has shared his experience as a Guest Editor for the Pertanika Journal of Science and Technology. He is involved in reviewing the process of different journals and conferences such as; IEEE transaction of computer, IET, IEEE conference on ITC, ADCOM, etc.

Prof. Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS). Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in Cloud Computing. He served as a Future Fellow of the Australian Research Council during 2012-2016. He has authored over 625 publications and seven textbooks, including "Mastering Cloud Computing" published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets, respectively. He has also edited several books, including "Cloud Computing: Principles and Paradigms" (Wiley Press, USA, Feb 2011). He is one of the highly cited authors in computer science and software engineering worldwide (hindex=116, g-index=255, 70,100+ citations). Microsoft Academic Search Index ranked Dr. Buyya as #1 author in the world (2005-2016) for both field rating and citations evaluations in the area of Distributed and Parallel Computing. "A Scientometric Analysis of Cloud Computing Literature" by German scientists ranked Dr. Buyya as the World's Top-Cited (#1) Author and the World's Most-Productive (#1) Author in the Cloud Computing. Recently, Dr. Buyya is recognized as a "2016 Web of Science Highly Cited Researcher" by Thomson Reuters and a Fellow of IEEE for his outstanding contributions to Cloud computing. For further information on Dr.Buyya, please visit his cyberhome: www.buyya.com.