HMM and Fuzzy Logic Based Algorithm for Efficient Task Scheduling and Resource Management in Cloud Systems

Vikash Goswami[#1], Dr. R.K. Shrivastava[*2]
[#]*Research Scholar, Deptt. of Mathematics,*
S.M.S Govt. Science College, Jiwaji University,
Gwalior, M.P. India
[*]*Professor, Deptt. of Mathematics,*
S.M.S Govt. Science College, Jiwaji University,
Gwalior, M.P. India

Abstract—*This paper presents a HMM and Fuzzy controller based combine approach for cloud incoming job queue prediction and managing VM status and configuration of VMs inside of cloud environment. The approach is aimed to efficiently serve the task requests with minimal resource and power utilization. The proposed technique uses HMM based approach to predict the job queue and applies it alongside with information of currently running VM's, VM's configurations, resource availability in the cloud, future jobs resource requirements, current job execution status etc. to a fuzzy logic based controller which then after controls the VM's status and configurations to serve the aimed purpose. The controlling in such way reduces the active physical resources which ultimately reduces the power requirements of the cloud. To validate the concept the proposed controller is tested against standard controlling algorithm for different load conditions. The test results obtained shows that the proposed fuzzy logic controller based technique outperforms standard techniques in terms of QoS, resource management, and power savings.*

Keywords —*Cloud Computing, Cloud VM Management, Fuzzy Logic Controller, HMM, Queue Management*.

I. INTRODUCTION

In a cloud, theuser services or applications runs over the Virtual Machines (VMs) formed inside the cloud system. The VMs as the name suggest, are formed by virtually allocating and configuringthe physical resources.The creation and management of these VMsare dome by virtualization platforms. Since cloud service provider who are bounded by the Service Level Agreement (SLA) has to maintain the minimum level Quality-of-Service (QoS), which also known as Service Level Objective (SLO).

The continuously growingadaption ofcloud systems, emerging new challenges for the cloud operators as they now have to handle larger loads of different operational and requirement characteristics with the limited resources. To handle such situations the cloud provider needed an efficient cloud managing algorithm that utilize its available resources in such a manner that it fulfills the user's requirements with guaranteed SLOwhile maintaining power requirements to lowest possible level.

This paper presents an adaptive task prediction with VM status and configuration management technique. The proposed technique uses HMM based prediction model for task prediction then this information is used by a fuzzy logic based decision making system to manage the status and configuration of the VMs.

The proposed technique is able to manage the balance among conditions such as task requirements, SLO bounds, available resources, power saving etc. by dynamically configuring the VMs. The organization of rest of the paper is as follows. The Section II providesa brief literature review. The Section III gives an overview of fuzzy logic controller. The Section IV explains the system architecture with simulation configurations,The Section Vexplains the proposed system. The simulation results are presented in Section VI andfinally, the Section VII, presents the conclusion and discussesthe possibilities of future works.

II. LITERATURE REVIEW

This section reviews the literature most related to our work. John J. Prevost et al. [19] introduce a novel framework integrating the load demand prediction and stochastic state transition models for optimal cloud resource allocation. The tradeoff is achieved between energy consumed and performance levels. The neural network and autoregressive linear prediction algorithms are used to predictthe loads in cloud datacenter.Zhenhuan Gong et al. [18] presented PredictiveElastic reSource Scaling (PRESS) scheme for cloud management.PRESS findsthe minute dynamic patterns from application resource demands and then use it for their resource allocations adjustment. The approach containslight weight signal processing and statistical learning algorithms to predict the dynamic application resource requirements.Sadeka Islam et al. [20] developeda prediction-basedresource estimation and provisioning methods using Neural Network and Linear

Regression to fulfillupcoming on-demand resource allocation in the cloud. In a cloud environment the task distribution is a process used to assigns the execution of tasks on distributedresources [6]. The most common way of doing that is using the optimization techniques which finds the optimal task to VM pair for given objective function and constrains. A number of techniques with different optimization techniques, objective functions and constrains have been proposed in [3, 4, 5, 8, 15, 16, and 17]. However all the techniques having optimization algorithm in common a number of difference in their utilization can be seen. For instance in [3] the Honey-Bee behavior (HBB) based optimization technique is used to achieve balancing of tasks over available VMs, the proposed algorithm manages the task execution priorities. The [5] and [15] both uses the particle swarm optimization (PSO), although the [5] adopted for deadline constrained task scheduling using self-adaptive learning, while [15] constrained task execution time and data transfer cost. In [8] a multi-objective genetic algorithm used. The multi-objective optimization provides extra facilities to achieve more than one objective simultaneously, hence it is able to achieve four different objectives, namely minimizing task transfer time, task execution cost, power consumption, and task queue length. In [16] the ant colony optimization is used to balance the entire system load at minimummakespan of a given tasksset. The improved differential evolution algorithm (IDEA) for the purpose in presented in [17], which combines the Taguchi method and a differential evolution algorithm(DEA).Their multi-objectiveoptimization approach uses thethe processing and receiving cost as the first objective and receiving, processing, and waiting time as second objective. Another approach for the task scheduling is using the fuzzy logic. The fuzzy logic is a method for solving complex decision making problem where the variables show some degree of overlapping. The fuzzy logic system has been successfully implemented for many control and decision making systems some of its application can be seen on washing machines, refrigerators and other household goods. In [12] the fuzzy logic is used for predictionof the virtual machine to assign the upcoming job, considering that the requirements of memory, bandwidth and disk space are imprecise. In [9] the taskscheduling model for virtual data centerswith uncertain workload and uncertain nodes availability is presented. The presented solution deals the problem as a two-objective optimizationas a trade-off betweenavailability and the average response time of VDC (virtual data center). Since the optimization requires the availability of VDC and workload values in advance the type-I and type-II fuzzy based predictor for VDC availability and Load-Balance is proposed. The hybrid job scheduling algorithm which involves genetic algorithm and fuzzy logic is presented in [1, 14]. The fuzzy logic is used here to reduce the number of iterations required by genetic algorithm to converse.

III. HIDDEN MARKOV MODEL (HMM)

3.1. Markov Models

In the Markov Model the prediction of the next state and its related observation onlydepends on the current state, or alternatively the state transition probabilities do not dependon the whole history of the past process [17]. This is called a first order Markov process the definition can be generalized for the i^{th}order Markov process as the probability of nextstate can be calculated by obtaining and taking account of the past i states. For the sequence of random variables$X = (X_I, \dots, X_r)$ takingvalues in some finite set $S = \{s_r, . \ . , s_r\}$, the state space. Then the MarkovProperties are [19]:

$$P(X_{t+1} = s_k | X_1, \dots \dots . X_t) = P(X_{t+1} = s_k | X_t), (limited\ hozrizon) (3.1)$$

$$= P(X_2 = s_k | X_1), (Time\ invariant) (3.2)$$

Because of the state transition is independent of time, we can have the following statetransition matrix A:

$$a_{ij} = P\left(X_{t+1} = s_j \middle| X_t = s_i\right), \dots \dots \dots \dots \dots . . (3.3)$$

a_{ij} is a probability, hence:

$$a_{ij} \geq 0, \forall i, j \sum_{j=1}^{N} a_{ij} = 1, \dots \dots \dots \dots . . (3.4)$$

Also we need to know the probability to start from a certain state, the initial statedistribution:

$$\pi_i = P(X_1 = s_i), where, \sum_{i=1}^{N} \pi_i = 1, . (3.5)$$

In a visible Markov model, the states from which the observations are produced and theprobabilistic functions are known so we can regard the state sequence as the output.

3.2. Hidden Markov Models

The Hidden Markov Model (HMM) extends the Markov Model for the cases where state knowledge is unavailable or in HMM, one does not know anything about what (system states) generates the observationsequence. The number of states, the transition probabilities, and from which state anobservation is generated are all unknown. Hence each state of the system is liked with observation with a probabilistic function instead of deterministic function as in case of Markov Model. At time t,an observation o_t is generated by a probabilistic function$b_j(o_t)$, which is associated withstate j, with the probability:

$$b_j(o_t) = P(o_t | X_t = j), \dots \dots \dots \dots \dots \dots (3.6)$$

3.3. Mathematical Terms in HMM

An HMM is composed of a five-tuple: (S, K, Π, A, B).

1. $S = \{1, \ldots, N\}$ is the set of states. The state at time t is denoted s_t.
2. $K = \{k_1, \ldots, k_M\}$ is the output observation and M is the number of observation choices.
3. Initial state distribution $\Pi = \{\pi_i\}, i \in S. \pi_i$ is defined as

$$\pi_i = P(s_1 = i), \ldots\ldots\ldots\ldots (3.7)$$

4. State transition probability distribution $A = \{a_{ij}\}, i, j \in S$.

$$a_{ij} = P(s_t + 1 | s_t), 1 \leq i, j \leq N, \ldots\ldots\ldots (3.8)$$

5. Observation symbol probability distribution $B = b_j(o_t)$. The probabilistic function for each state j is:

$$b_j(o_t) = P(o_t | s_t = j), \ldots\ldots\ldots\ldots\ldots (3.9)$$

After modeling a problem as an HMM, and assuming that some set of data was generated by the HMM, we are able to calculate the probabilities of the observation sequence and the probable underlying state sequences. Also we can train the model parameters based on the observed data and get a more accurate model. Then use the trained model to predict unseen data.

To generate the HMM model for any system we need to compute three parameters

1. Observation Sequence Computing: The probability of the observation sequences.
2. The state sequence $(1, \ldots, N)$ that best "explains" the observations.
3. The tuning of the parameters to find the best model for given observation sequence O, and a space of possible models.

Since in this work we only used the probability of the observation sequences hence the second and third terms are not explained here.

3.3.1 Finding the probability of an observation

Given an observation sequence $O = (o_1, \ldots, o_T)$ and an HMM $\lambda = (A, B, \Pi)$, we want to find out the probability of the sequence $P(O|\lambda)$. This process is also known as decoding. Since the observations are independent of each other, the probability of a state sequence $S = (s_1, \ldots, s_T)$ generating the observation sequence can be calculated as:

$$P(O|\lambda) = \sum_S P(O|S, \lambda) P(S|\lambda), \ldots\ldots\ldots\ldots (3.10)$$

$$= \sum_{s_1, \ldots\ldots s_{T+1}} \pi_{s_1} \prod_{t=1}^{T} a_{s_t s_{t+1}} b_{s_t s_{t+1} o_t} \ldots\ldots\ldots\ldots (3.11)$$

The computation is quite straightforward by summing the observation probabilities for each of the possible state sequence.

IV. FUZZY LOGIC SYSTEM

Fuzzy logic is an approach where a variable simultaneously belongs to more than one class with certain degree and the degree of membership is defined by membership function. The fuzzy logic approximate decision making using natural language terms. It is especially useful in modeling of systems where information cannot be defined precisely, but some broad definitions can be formed. Because of its simplicity and effectiveness, Fuzzy-logic technology has gained many applications in scientific and industrial applications.

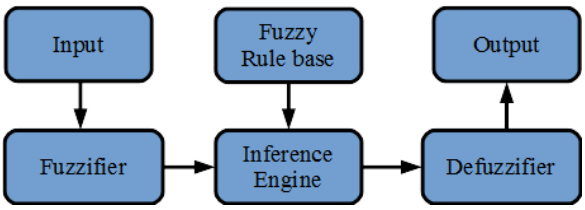


Figure 1: Typical architecture of Fuzzy Logic Controller (FLC) system.

A typical architecture of FLC is shown in fig. 1, which comprises of four principal comprises: a Fuzzifier, a Fuzzy rule base, inference engine, and a de-Fuzzifier.

Fuzzification: The Fuzzification is the process of converting crisp values in terms of degree of membership with different classes. The degree of membership is calculating using membership functions. The Fuzzification enables variable association with linguistic term.

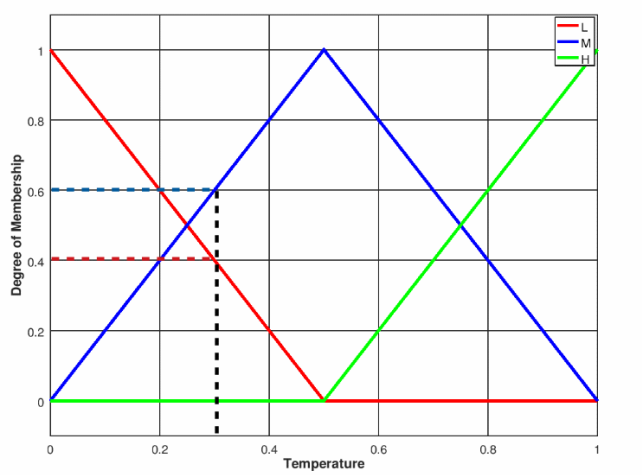


Figure 2: Showing Fuzzification process with triangular membership function.

The fig. 2, shows the mapping the crisp value of temperature with L, M and H categories, the figure shows that for 0.3 task priority (black dashed line) it has the membership of 0.4 (red dashed line), 0.6 (blue dashed line) and 0 for categories L, M and H respectively. In fig. 2 the membership function is of triangular however it can be, trapezoidal, Gaussian, bell-shaped, sigmoidal etc. The exact type depends on the actual application.

Fuzzy Rule Base: this contains the rule which relates the fuzzy inputs and outputs in linguistic terms. It contains a sequence of the form IF-THEN. For example in a heating system it can be defined as

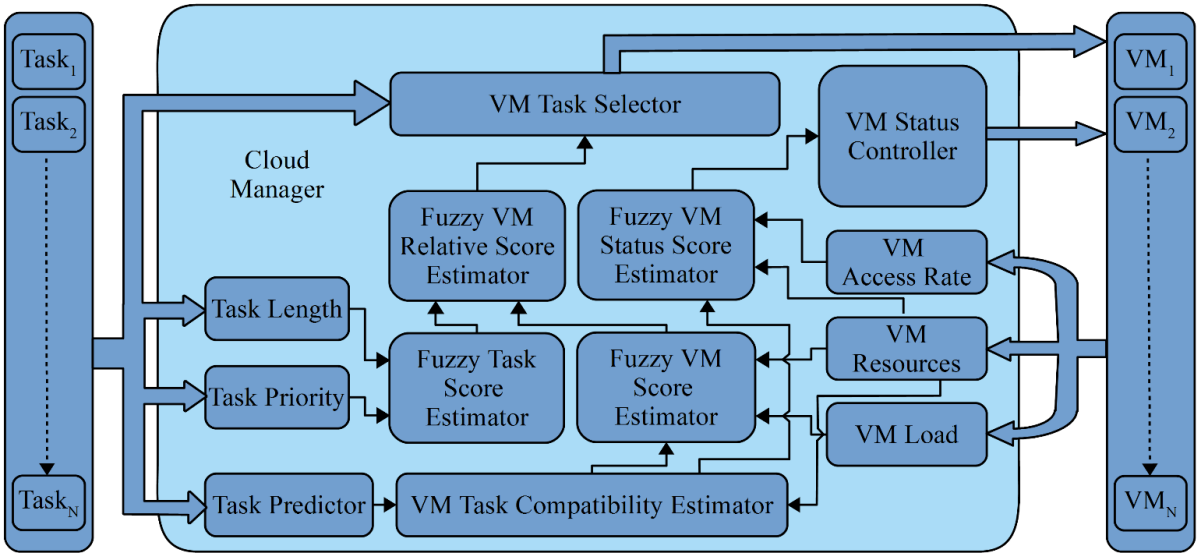
IF temperature is L THEN set heater power to H
IF temperature is M THEN set heater power to M
IF temperature is H THEN set heater power to L

Defining these rules are requires expertize in related field of application.

Inference Engine: It executes all the fuzzy rules available in the fuzzy rule base for the available inputs. This produces a number of fuzzy outputs one for each rule.

De-Fuzzification: The output from inference engine is still fuzzy which must be converted into crisp value before it can be used with any non-fuzzy system. This conversion of fuzzy outputs to crisp value is done by De-Fuzzification. The De-Fuzzification can be performed by many ways such as using Centroid, Max-Membership, Weighted Average and Mean-Max Membership etc.

Figure 3: The proposed system architecture showing the different functional blocks and their interconnections.



V. PROPOSED ALGORITHM

5.1. System Architecture

The architecture of the proposed system is presented in fig.3 the system contains four fuzzy logic decision making blocks, two VM controlling blocks and five information extraction blocks. The working details of each block is as follows:

- ***Information extraction blocks:***these blocks are used to extract useful information from task queue and VMs.
  - Task Length: the length of current task in MIPS.
  - Task Priority: Execution priority of current task.
  - Task Predictor: Estimates the probability of arriving of different tasks.
  - VM Access Rate: how many times the particular VM has accessed during predefined time interval.
  - VM Resources: the resources used by VM.
  - VM Load: current load on VM.

Task Predictor:this block predicts the upcoming tasks in the cloud. This prediction is done using dual HMM model where one HMM is used to calculate the probability of generating request by any user in the upcoming sampling window and the second HMM for calculating the request pattern that may be generated by any user in the upcoming sampling window. After that these two results are combined to estimate the exact request pattern in the upcoming sampling window.

For the modeling we divided the tasks into S different task categories and the task request arriving in the cloud is replaced by the category symbol closely related to it. This limits the total number of observations to S. The length of the sampling window W is the length of sequences of symbols used to predict the upcoming symbols sequences.

Let the task requests in the cloud for some sampling window length be $R = \{r_1^a, r_2^b, r_3^c, r_4^b \dots\dots\dots r_W^s\}$, where $a, b, c, \dots s \in N,\ r \in S$ and r_1^a is representing the 1^{st} entry of request in the sampling window generated by a^{th} user.

The N number of different users given as $U = \{u_1, u_2, \dots\dots\dots\dots\dots u_N\}$.

Let the last Ltask requests generated by user u_i is defined as $s_i = \{r_{-1}^i, r_{-2}^i, \dots\dots\dots r_{-L}^i\}$, and the task requested by i^{th} user appeared in M previous sampling windows is defined as$g_i = \{b_{-1}^i, b_{-2}^i, b_{-3}^i, \dots\dots, b_{-M}^i\}, b \in \{0,1\}$, the b is a binary variable and b_m^i states that there was at least one request in the m^{th} previous sampling window generated by i^{th} user.

Now using the equation described in section 3.10the probability of requesting a specific symbol by a specific user at the next W events and the probability of generating the request by each user in upcoming sampling windows can be calculated.

Let the calculated probability of arriving of each task request (symbol) in upcoming sampling window by i^{th} user be$Pr_i = \{pr_1^i, pr_2^i, pr_3^i, \dots\dots\dots, pr_N^i\}$, and the probability of generating any task request (symbol) in the next sampling window by i^{th} user bet_i.

The total probability of appearing a task request r_j in the upcoming sampling windows is calculated as

$$Pr_i^{total} = \sum_{i=1}^{N} t_i \cdot pr_j^i$$

Hence the total probability of arriving of each task request in upcoming sampling window can be given as

$$Pr^{total} = \{Pr_1^{total}, Pr_2^{total}, Pr_3^{total}, \ldots., Pr_S^{total}\}$$

Since Pr^{total} is normalized we need to convert it into normalized probability by calculating Pr_{all}^{total} and diving Pr^{total} by it as follows

$$Pr_{all}^{total} = \sum_{i=1}^{S} Pr_i^{total}$$

$$P_{norm}^{total} = \{\frac{Pr_1^{total}}{Pr_{all}^{total}}, \frac{Pr_2^{total}}{Pr_{all}^{total}}, \frac{Pr_3^{total}}{Pr_{all}^{total}}, \ldots., \frac{Pr_S^{total}}{Pr_{all}^{total}}\}$$

$$P_{norm}^{total} = \{P_1, P_2, P_3, \ldots\ldots\ldots\ldots\ldots\ldots., P_S\}$$

The P_{norm}^{total} is an array that contains the normalized probability of all the tasks in the upcoming sampling window.

- ***VM Task Compatibility Estimator Block:*** this block tests capability of VMs for executing the predicted tasks within given time bound. The higher value of this estimator for given task-VM pair indicates the faster execution of task by that VM. However higher values pair are not the most preferable because it may cause poor resource utilization. The estimation is done as follows:

$$S_{TC}^{i,j} = \begin{cases} 0, & if \left(\frac{TL_j^P + L_C^i}{C_{VM}^i} \leq \frac{1}{TP_j^P}\right) \\ \frac{(TL_j^P + L_C^i)}{C_{VM}^i} P_j, & otherwise \end{cases}$$

  Where $S_{TC}^{i,j}$ presents the VM Task Compatibility Score of the i^{th} available VM for the j^{th} predicted task. Other terms used are defined as follows:

  VM's Execution Capacity of the i^{th} available VM = C_{VM}^i
  VM's Current Load on the i^{th} available VM = L_C^i
  Predicted Task's Priority for the j^{th} task = TP_j^P
  Predicted Task's Length for the j^{th}task = TL_j^P
  Task Arriving Probability for the j^{th} task = P_j

- ***Fuzzy Logic Decision Making Blocks:*** these blocks are used to make specific decisions based on provided inputs using fuzzy logic.
  - Fuzzy Task Score Estimator: this block estimates the task requirements on the basis of task length and priority.
  - Fuzzy VM Score Estimator: this block estimates the VM capability to handle tasks on the basic of VM configuration and load.
  - Fuzzy VM Relative Score Estimator: this block estimates fitness between VM capability and task requirements.

- Fuzzy VM Status Score Estimator: this block is used to decide the operational status of VM on the basis of resource utilized by VM and access rate of VM.

- VM Task Selector: this block is used to select the best VM for the current task, it takes the input from Fuzzy Relative Score Estimator for all the VMs and selects the VM having the highest relative score.

- VM status Controller: it maintains the states of all VM on the basis of VM status score (VM_{score}). This blocks uses two different thresholdst_{sleep} and $t_{dissolve}$ where $0 < t_{sleep} < t_{dissolve} < 1$ which are compared against VM status score to decide the VM status as follows:

$$VM_{status} = \begin{cases} Keep\ Running, & if\ VM_{score} < t_{sleep} \\ Sleep, & elseif\ t_{sleep} \geq VM_{score} > t_{dissolve} \\ Dissolve, & else\ VM_{score} \geq t_{dissolve} \end{cases}$$

Fuzzy Membership Function Selection: the selection of member function in most difficult and important task for any fuzzy system. Because these membership function defines the fuzziness and the way variable changes their memberships to different classes, the improper selection of these function can drastically degrade the performance of fuzzy system. In the proposed work the triangular member
-ship function (as shown in fig. 4(a)) is used for the variables Task Length, VM Load, VM resources and VM task compatibility score, the triangular membership function is selected for these variable because these parameters are consider to have the linear transition in degree of membership from one to another class. The two sided Gaussian membership function (as shown in fig. 4(b)) is used for the Access Rate, because we want to increase its importance much quickly then linear rate. The two sided Gaussian membership function is designed as its transition from present class to higher class is much fasterand it also leaves it present class much sharply. The Gaussian membership function (as shown in fig. 4(c)) is used for variable Task Priority, as the priority is considered to have a continuous symmetric transition.

5.2. Model Terminology

Before going to description this section explains the terminology used in the algorithm.

Task Length(TL_i) = the length of the i^{th} task.

Execution Capacity of VM(C_{VM}^i) = rate of task execution for i^{th} VM.

Current Load on VM(L_c^i) = the remaining task length of currently executing task on i^{th} VM.

Access Rate of VM(R^i) = number of different tasks assigned to i^{th} VM per unit time.

Predicted Task's Priority for the j^{th} task =TP_j^P.

Predicted Task's Length for the j^{th}task =TL_j^P.

Task Arriving Probability for the j^{th} task =P_j.

Task Waiting Time(T_w^i) = describes the time only after that the i^{th} VM can start execution of the requested task. It can also be described as the time required to finish the currently executing tasks on the i^{th} VM.

$$T_w^i = L_c^i / C_{VM}^i$$

Task Execution Time($T_{e,j}^i$) = the time required by the i^{th} VM to execute the j^{th} task.

$$T_{e,j}^i = TL_j / C_{VM}^i$$

Total Task Completion Time($T_{t,j}^i$) = it is the sum of Task Waiting Time and Task Execution Time.

$$T_{t,j}^i = T_w^i + T_{e,j}^i$$

Task Priority (TP_i) = is the inverse of required maximum Total Task Completion Timefor the i^{th} task.

VM Task Compatibility Score $S_{TC}^{i,j}$ = Presents the VM Task Compatibility of the i^{th} available VM for the j^{th} predicted task.

$$S_{TC}^{i,j} = \begin{cases} 0, & if\left(\frac{TL_j^P + L_C^i}{C_{VM}^i} \leq \frac{1}{TP_j^P}\right) \\ \frac{(TL_j^P + L_C^i)}{C_{VM}^i} P_j, & otherwise \end{cases}$$

$$S_{TC}^{i,All} = \sum_{j=1}^{M} S_{TC}^{i,j}$$

$S_{TC}^{i,All}$ = is the Total VM Task Compatibility Score of i^{th} VM for all the predicted tasks.

Fuzzy Task Score Estimator =$F_{TS}(TL_i, TP_i)$.

Fuzzy Task Score(S_{TS}^i) = task score of i^{th} task calculated by Fuzzy Task Score Estimator$F_{TS}(\quad)$.

Fuzzy VM Score Estimator =$F_{VM}(L_c^i, C_{VM}^i, S_{TC}^{i,All})$.

Fuzzy VM Score(S_{VM}^i) = VM score of i^{th} VM calculated by Fuzzy VM Score Estimator$F_{VM}(\quad)$.

Fuzzy VM Relative Score Estimator =$F_{VMR}(S_{TS}^i, S_{VM}^i)$.

Fuzzy VM Relative Score(S_j^i) = relativeVM score of i^{th} VM forj^{th} task calculated by Fuzzy VM Relative Score Estimator$F_{VMR}(\quad)$.

Fuzzy VM Status Score Estimator =$F_{SS}(R_c^i, C_{VM}^i, S_{TC}^{i,All})$.

Fuzzy VM Status Score (S_{SS}^i) = VM status score of i^{th} VM calculated by Fuzzy VM Status Score Estimator$F_{VM}(\quad)$.

5.3. Proposed Algorithm

Let the task length and priority of newly arrived task be TL_{new} and TP_{new} respectively.

$$N = total\ numbers\ of\ VMs\ currently\ running\ in\ VM.$$

$M = Categories\ of\ Tasks.$

$W = Sampling\ Window\ Length.$

$U = Total\ Numberof\ Users.$

Start Main Routine
PredictTasks ( );
for $i = 1$ *to* N
$$T_w^i = \frac{TL_{new}}{C_{VM}^i}$$
$$T_{e,j}^i = \frac{TL_j}{C_{VM}^i}$$
$$T_{t,j}^i = T_w^i + T_{e,j}^i$$
endfor
$VM_{available} = 0$
for $i = 1$ *to* N
$$if\ (T_{t,j}^i > \frac{1}{TP_{new}})$$
$VM_{available} = 1$
break ;
endif
endfor
if ($VM_{available} == 0$)
CreateNewVM ( );
else
AssignTask ( );
endif
End Main Routine

Start Sub − Routine CreateNewVM
$C_{VM}^{new} = TL_{new} \times TP_{new}$;
create new VM_{new} *with configuration* C_{VM}^{new} ;
assign Task to VM_{new} .
End Sub − Routine CreateNewVM

Start Sub − Routine AssignTask
$S_p = 0$;
$VM_{selected} = 0$;
for $i = 1$ *to* N
$S_{TC}^{i,All} = 0$;
for $j = 1 : M$
$S_{TC}^{i,All} = S_{TC}^{i,All} + S_{TC}^{i,j}(L_c^i, C_{VM}^i, TL_j^P, TP_j^P, P_j)$;
end
$S_{VM}^i = F_{VM}(L_c^i, C_{VM}^i, S_{TC}^{i,All})$;
$S_{TS}^i = F_{TS}(TL_i, TP_i)$;
$S_c^i = F_{VMR}(S_{TS}^i, S_{VM}^i)$;
if($S_c^i > S_p$)
$S_p = S_c^i$;
$VM_{selected} = i$;
endif
endfor

$assign\ Task\ to\ VM_{selected}$.

End Sub – Routine AssignTask

Start Sub – Routine VMStatusControl

for $i = 1\ to\ N$

if $(isIdle(VM_i))$

$S_{TC}^{i,All} = 0;$

for $j = 1{:}M$

$S_{TC}^{i,All} = S_{TC}^{i,j}(L_c^i, C_{VM}^i, TL_j^P, TP_j^P, P_j);$

end

$S_{SS}^i = F_{SS}(R_c^i, C_{VM}^i, S_{TC}^{i,All});$

if $(S_{SS}^i > TH_{dissolve})$

$Dissolve\ the\ VM_i;$

elseif $(S_{SS}^i > TH_{sleep})$

$Set\ VM\ to\ Sleep$

else

$do\ nothing$

endif

endif

endfor

End Sub – Routine VMStatus Control

$s_i = \{r_{-1}^i, r_{-2}^i, \ldots\ldots\ldots r_{-L}^i\}$; //(last Lrequests by i^{th} user.)

$g_i = \{b_{-1}^i, b_{-2}^i, b_{-3}^i, \ldots\ldots, b_{-M}^i\}, b \in \{0,1\}$ // requests by i^{th} user in M // previous sampling windows.

$P_i = \{p_1^i, p_2^i, p_3^i, \ldots\ldots\ldots, p_N^i\}$; // the probability of request of all tasks by

// i^{th} user at the next W events.

// using the equation described

// in section 3.10

$T_i = \{t_1, t_2, t_3, \ldots\ldots t_U\}$; // the probability of generating the request by

// each user in upcoming

// samplingwindows.

// in section 3.10

$P_i^{total} = \sum_{i=1}^{N} t_i \cdot p_j^i$;//The total probability of appearing a request r_j in

//

upcomingsamplingwindows.

$P^{total} = \{P_1^{total}, P_2^{total}, P_3^{total}, \ldots\ldots, P_S^{total}\}$; // the total probability of

// arriving of each request in

// upcoming sampling window.

$P_{all}^{total} = \sum_{i=1}^{S} P_i^{total}$;

$P_{norm}^{total} = \left\{\frac{P_1^{total}}{P_{all}^{total}}, \frac{P_2^{total}}{P_{all}^{total}}, \frac{P_3^{total}}{P_{all}^{total}}, \ldots\ldots, \frac{P_S^{total}}{P_{all}^{total}}\right\}$; // normalized probability

$P_{norm}^{total} = \{P_{1,norm}^{total}, P_{2,norm}^{total}, P_{3,norm}^{total}, \ldots\ldots, P_{S,norm}^{total}\}$;

5.4. Proposed Algorithm Explanations

As shown in fig. 4 the cloud manager waits for the arrival of the new task and as soon as it receives the new task from the task queue it extracts the task related parameters like task length and task priority. On the other process it estimates the upcoming tasks length and priority, which are required to estimate the task VM compatibility score.

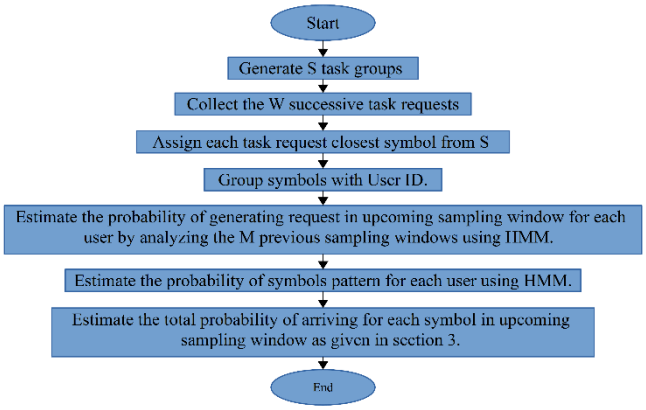


Table 1: The Fuzzy Rules and Surf plots for all fuzzy controllers.

Figure 5.1: Flow chart of the proposed algorithm.

The VM compatibility score is used as the possibility of VMs utilization in future. Once it founds these values it checks all the VMs for condition $T_{t,j}^{i} > \frac{1}{TP_{new}}$(as shown in algorithm), if it did not find any VM then it move to create a new VM according to the task requirements (as shown in algorithm sub-routine *AssignTask*). Otherwise if it find then calculate the relative VM score for all such VMs using fuzzy rules defined in table 1(c). to estimate VM relative score it send task length and priority values to fuzzy task score estimator (as shown in fig. 3). This fuzzy estimator calculates the score according to the values and the rules defined in table 1(a). The fuzzy task score works as one input for the fuzzy relative score estimator for the second input the cloud manager scans all the running VMs for their execution capacity and current load. The above two values in then applied to fuzzy VM score

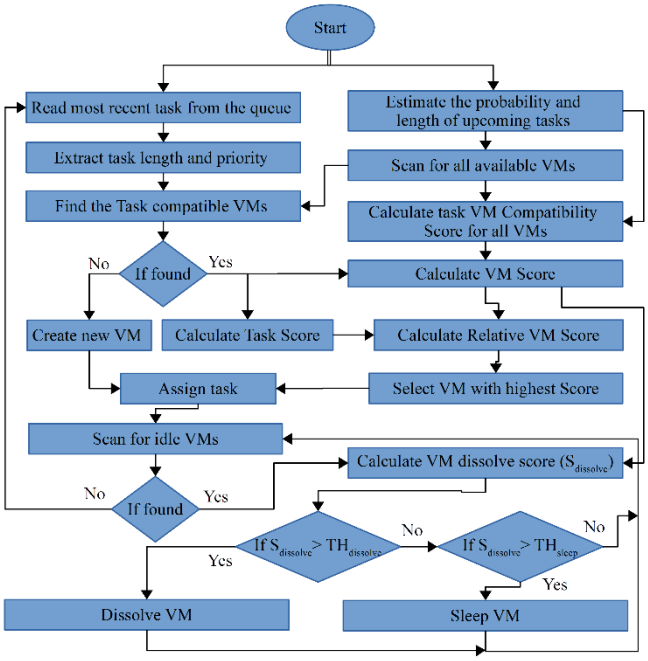


Figure 5.2: Flow chart of the proposed algorithm.

Estimator block which estimates the VM score according to the rules defined in table 1(b). The VM score works as second input for the VM relative score estimator. The procedure is repeated for all the running VMs and the relative scores are stored. Now the VM with highest relative score is selected for assignment of input task.

To manage the status of VMs the cloud manager scans each VM for their execution capacity, accessing rate and VM compatibility score then these values are used to estimate their status score by applying fuzzy rules defined in table 1(d). The calculated status score is compared against $TH_{dissolve}$ and if it finds VM score greater then it dissolves the VM and reclaim its resources. Otherwise it check the score against TH_{sleep} to check if it can be set into sleep state or should keep running.

Fuzzy Task Score Estimator		Task Length				
		VL	L	M	H	VH
Task Priority	VL	*VL*	*VL*	*L*	*L*	*M*
	L	*VL*	*L*	*L*	*M*	*M*
	M	*L*	*L*	*M*	*M*	*H*
	H	*L*	*M*	*M*	*H*	*H*
	VH	*M*	*M*	*H*	*H*	*VH*

1(a)

Fuzzy VM Rel. Score Estimator		Task Score				
		VL	L	M	H	VH
VM Score	VL	*M*	*L*	*VL*	*VL*	*VL*
	L	*M*	*M*	*L*	*VL*	*VL*
	M	*H*	*M*	*M*	*L*	*VL*
	H	*VH*	*H*	*M*	*M*	*L*
	VH	*VH*	*VH*	*H*	*M*	*M*

1(b)

Fuzzy VM Score Estimator		VM Resources				
		VL	L	M	H	VH
VM Load	VL	*L*	*M*	*M*	*H*	*H*
		M	*H*	*H*	*VH*	*VH*
		M	*M*	*M*	*M*	*H*
	L	*VL*	*L*	*L*	*M*	*M*
		L	*M*	*M*	*H*	*H*
		M	*M*	*M*	*M*	*M*
	M	*VL*	*VL*	*L*	*L*	*M*
		VL	*L*	*M*	*M*	*H*
		L	*M*	*L*	*M*	*M*
	H	*VL*	*VL*	*VL*	*L*	*L*
		VL	*VL*	*L*	*M*	*M*
		L	*L*	*M*	*M*	*M*
	VH	*VL*	*VL*	*VL*	*VL*	*L*
		VL	*VL*	*VL*	*L*	*M*
		L	*L*	*L*	*M*	*M*

1(c)

Fuzzy VM Status Score Estimator		Task Access Rate				
		VL	L	M	H	VH
VM Resources	VL	*L*	*L*	*VL*	*VL*	*VL*
		M	*M*	*L*	*VL*	*VL*
		M	*M*	*M*	*L*	*L*
	L	*M*	*L*	*L*	*VL*	*VL*
		H	*M*	*M*	*VL*	*VL*
		H	*M*	*M*	*L*	*L*
	M	*M*	*M*	*L*	*VL*	*VL*
		H	*H*	*M*	*L*	*VL*
		H	*M*	*M*	*M*	*L*
	H	*H*	*M*	*M*	*L*	*VL*
		VH	*H*	*H*	*M*	*L*
		H	*H*	*M*	*M*	*M*
	VH	*H*	*H*	*M*	*M*	*L*
		VH	*VH*	*H*	*H*	*M*
		VH	*H*	*M*	*M*	*M*

1(d)

The colors In rows of table (c) and (d) represents the Low (L), Medium (M) and High (H) values of variable Task VM Compatibility Score.

VI. SIMULATION RESULTS

6.1. Numerical Model Considerations for Cloud System

1. It is assumed that the load balancer knows the configuration (like processing capacity, memory etc.) of each virtual machine (VM) in the cloud.
2. The load balancer can get the operational state of each VM with zero time delay.
3. The load balancer takes no time in selecting and assigning the tasks to VM's.
4. The load balancer selects the VM for the input tasks on the basis of selected algorithm.
5. Each VM has zero booting time hence start executing assigned task immediately.
6. The incoming tasks size is considered in MI (million instructions) units.
7. The VM's capacities are also considered in MIPS (million instructions per second) units.

The evaluation of the proposed algorithm is performed using OCTAVE/ MATLAB numerical computing software. During the simulation the tasks arrive as a Poisson process at a rate of λ. The random length tasks within the provided minimum and maximum task lengthlimits aregenerated using a uniform discrete distribution. The similar way is used for the generation of task priorities and defining the VM execution capacities.

6.2. Definition of Evaluation Terms

The following measures are used to evaluate the performance of the algorithm.

SLAFailure: is defined as failure of the cloud in serving the task within given time bound (inverse of priority).

SLA Failure Task Length: defines the length of the *SLAFailure*task.

VM Reboots: is the booting of VMs from sleep mode, this operation is required when the already running VMs cannot serve the current task.

VM Reforms: is the formation of new VM form the available unused resources when the current task cannot be handled by the already formed (running or sleeping) VMs.

Resource Utilization Efficiency: is presents that how efficiently the cloud resources are utilized to serve the tasks, and it is calculated as follows:

$$Resource\ Utilization\ Efficiency\ = \frac{\sum_{i=1}^{N} TL_i}{\sum_{i=1}^{N}\left(\sum_{j=1}^{A_i} C_{VM}^{j}\right)} \times 100$$

Where TL_i: is the load in cloud at time i.

A_i: is the number of VMs active and running at time i.

C_{VM}^{j}: Execution capacity of the j^{th} VM.

N: is the total simulation time (discrete events of task arrival).

5.3Simulation Environment Configurations

To simulate the algorithm properly some important parameters are required to configure these parameters and their values are listed in table 2.

Table 2: The simulation parameters and their values.

Configuration Parameter	Parameter Value
Total Execution Capacity Available	100 MIPS
Sampling Window Length	10
Minimum Task Length	100 MI
Minimum Task Execution Time	1 Seconds
Maximum Task Execution Time	10 Seconds
Threshold Sleep	0.5
Threshold Dissolve	0.7
Total Simulation Time	100 Seconds

6.3. Simulation Outcomes

The outcomes of the simulation is presented in graphical forms. The outcomes of the proposed algorithm is also compared with the two standard task scheduling algorithm names Round Robin and Random Selection.

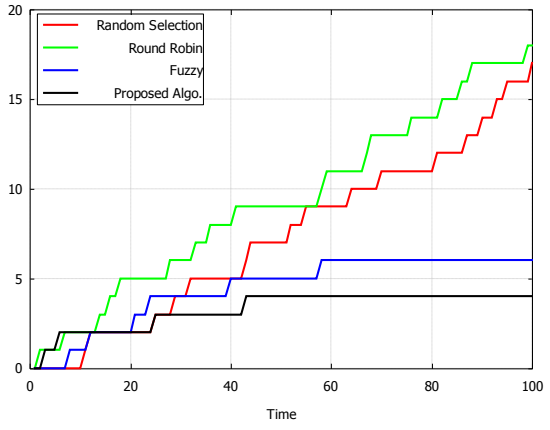


Figure 6: Plot for number of tasks failed to receive the requested SLA by cloud due shortage of resources with respect to simulation time.

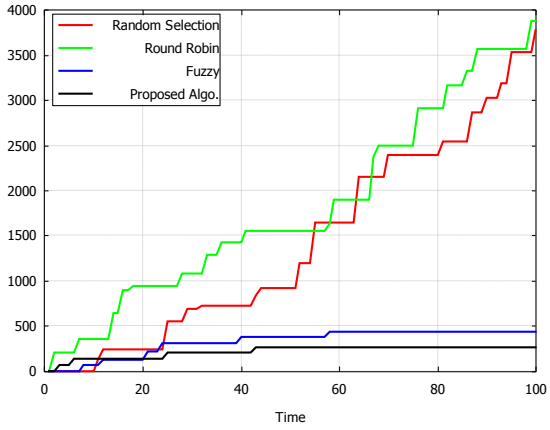


Figure 7: Plot for total length of the tasks which failed to receive the requested SLA by cloud due shortage of resources with respect to simulation time.

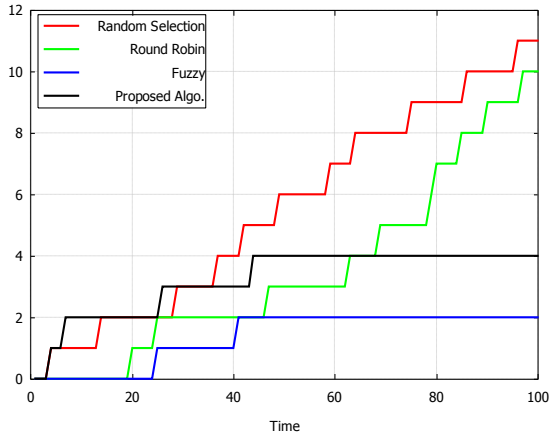


Figure 8: Plot for number of time the VM is rebooted from the sleep mode for assignment of task with respect to simulation time.

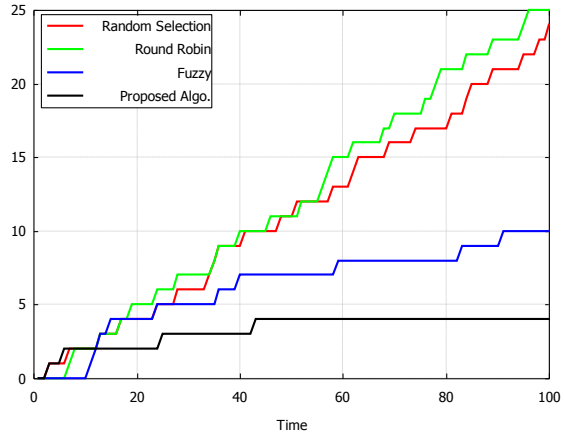


Figure 9: Plot for number of time the VM is reformed from the available resources for assignment of task with respect to simulation time.

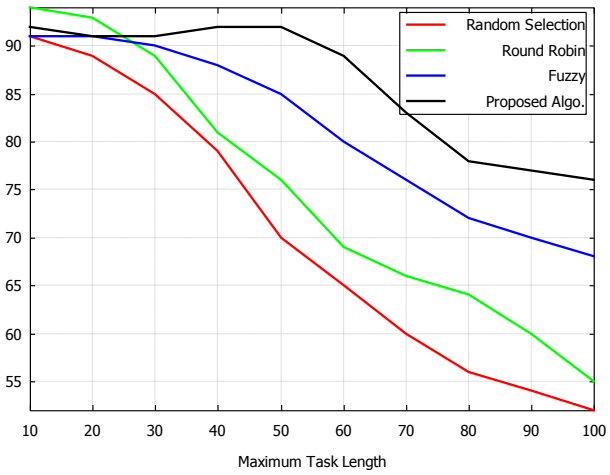


Figure 10: Plot showing the variation of the cloud resource utilization efficiency with respect to maximum task length.

VII. CONCLUSION

In this paper, we presented the HMM andfuzzy logic based task scheduling and resource management scheme for Cloud systems. The simulation results shows that the proposed algorithm reduces the number of tasks and total tasks length, cloud failed to deliver the guaranteed SLA by 35% and 50% respectively when compared to fuzzy based algorithm while compare to the conventional algorithms reduce by a factor of 4.0 (fig. 6) and 8.0 (fig. 7). This shows the algorithm rejects tasks with lowest length when it failed to deliver guaranteed SLA.

The proposed algorithm show greater number of VM reboots (fig. 8) although this causes delay but the algorithm compensate this delay by reducing the number of new VM formation (fig. 9). Since the formation of new VM take many time larger time than rebooting the overall delay remain lower than the fuzzy based algorithm.

Finally the comparison of efficiency (fig. 10) shows that the proposed algorithm gives the maximum efficiency when operated at moderate load conditions, which is appreciable as mostly cloud operates at such conditions. It also shows that the efficiency falls much slowly than the other algorithm, hence it can be said that the algorithm provides much uniform and stable performance for a wide range of loading conditions.

These results depicts that the proposed algorithm adequately handles the task scheduling and resource management of a cloud system having limited resources and SLA bounds.
Although the presented algorithm provide better results than the conventional algorithms compared in this paper. The proposed algorithm can be further improved by fine tuning the membership functions and rule base however these modifications are leaved for the future work.

REFERENCES

[1] M. Shojafar, S. Javanmardi, S. Abolfazli, and N. Cordeschi, "Erratum to: FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method," Cluster Computing, vol. 18, no. 2, pp. 845–845, 2015..

[2] H. Chen, F. Wang, N. Helian, and G. Akanmu, "User-priority guided Min-Min scheduling algorithm for load balancing in cloud computing," 2013 National Conference on Parallel Computing Technologies (PARCOMPTECH), 2013.S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A novel ultrathin elevated channel low-temperature poly-Si TFT," *IEEE Electron Device Lett.*, vol. 20, pp. 569–571, Nov. 1999.

[3] M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, "High resolution fiber distributed measurements with coherent OFDR," in *Proc. ECOC'00*, 2000, paper 11.3.4, p. 109.

[4] C.-W. Tsai and J. J. P. C. Rodrigues, "Metaheuristic Scheduling for Cloud: A Survey," IEEE Systems Journal, vol. 8, no. 1, pp. 279–291, 2014.

[5] X. Zuo, G. Zhang, and W. Tan, "Self-Adaptive Learning PSO-Based Deadline Constrained Task Scheduling for Hybrid IaaS Cloud," IEEE Transactions on Automation Science and Engineering, vol. 11, no. 2, pp. 564–573, 2014.

[6] S. Adabi, A. Movaghar, and A. M. Rahmani, "Bi-level fuzzy based advanced reservation of Cloud workflow applications on distributed Grid resources," The Journal of Supercomputing, vol. 67, no. 1, pp. 175–218, 2013.

[7] D. Poola, S. K. Garg, R. Buyya, Y. Yang, and K. Ramamohanarao, "Robust Scheduling of Scientific Workflows with Deadline and Budget Constraints in Clouds," 2014 IEEE 28th International Conference on Advanced Information Networking and Applications, 2014.

[8] F. Ramezani, J. Lu, J. Taheri, and F. K. Hussain, "Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments," World Wide Web, vol. 18, no. 6, pp. 1737–1757, Oct. 2015.

[9] X. Kong, C. Lin, Y. Jiang, W. Yan, and X. Chu, "Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction," Journal of Network and Computer Applications, vol. 34, no. 4, pp. 1068–1077, 2011.

[10] W. Lin, C. Liang, J. Z. Wang, and R. Buyya, "Bandwidth-aware divisible task scheduling for cloud computing," Software: Practice and Experience, vol. 44, no. 2, pp. 163–174, 2012.

[11] S. K. Garg, A. N. Toosi, S. K. Gopalaiyengar, and R. Buyya, "SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter," Journal of Network and Computer Applications, vol. 45, pp. 108–120, 2014.

[12] M. S. Q. Z. Nine, M. A. K. Azad, S. Abdullah, and R. M. Rahman, "Fuzzy logic based dynamic load balancing in virtualized data centers," 2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2013.

[13] A. N. Toosi and R. Buyya, "A Fuzzy Logic-Based Controller for Cost and Energy Efficient Load Balancing in Geo-distributed Data Centers," 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), Limassol, 2015, pp. 186-194.

[14] Javanmardi S., Shojafar M., Amendola D., Cordeschi N., Liu H., Abraham A. "Hybrid Job Scheduling Algorithm for Cloud Computing Environment," IBICA 2014. Advances in Intelligent Systems and Computing, vol 303. Springer, Cham.

[15] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments," 2010 24th IEEE International Conference on Advanced Information Networking and Applications, 2010.

[16] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization," 2011 Sixth Annual Chinagrid Conference, 2011.

[17] J.-T. Tsai, J.-C. Fang, and J.-H. Chou, "Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm," Computers & Operations Research, vol. 40, no. 12, pp. 3045–3055, 2013.

[18] Z. Gong, X. Gu, and J. Wilkes, "PRESS: PRedictive Elastic ReSource Scaling for cloud systems," 2010 International Conference on Network and Service Management, 2010.

[19] J. J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, "Prediction of cloud data center networks loads using stochastic and neural models," 2011 6th International Conference on System of Systems Engineering, 2011.

[20] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," Future Generation Computer Systems, vol. 28, no. 1, pp. 155–162, 2012.