Load Balancing based on Statistical Model in Expert Cloud

Shiva Razzaghzadeh[1], Ahmad Habibizad Navin[2*], Amir Masoud Rahmani[1], Mehdi Hosseinzadeh[1]
1- Department of Computer engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran.
Email: shiva.razzaghzadeh@srbiau.ac.ir
Email: rahmani@srbiau.ac.ir
Email: hosseinzadeh@srbiau.ac.ir
2- Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran.
Email: a.habibizad@srbiau.ac.ir (Corresponding author)

Received: January 2019 Revised: March 2019 Accepted: June 2019

ABSTRACT:
Expert Cloud is a new class of cloud computing which enables the users to achieve their requirements from a collection containing experts and skills created by the Human Resources (HRs). The acquisition of these skills and experts from this collection is possible by using the internet and cloud computing concepts without consideration of the HRs location. The load balancing in cloud computing means equal load distribution among resources, Virtual Human Resources (VHRs) and servers. The effective load distribution in a heterogeneous environment such as cloud is an important challenge. The increase in the number of users, the differences of request types and also different resources capabilities and capacities cause that some resources become overload and some others become idle. This paper presents a dynamic load balanced task scheduling algorithm in expert cloud. In this method, we utilize the Genetic Algorithm (GA) as a ranking for making distinction among the VHRs capabilities. In the proposed method, interval estimation and specification matrix are used to allocate the VHRs and also to determine the service rate. The load balancing and mapping process are modeled based on Simple Exponential Smoothing and Probability Theory. This statistical load balancing model allows allocating the VHRs based on service rate and Poisson model. Thus, each task is delivered to the VHR; which is capable to execute it. The simulation results have shown that the expert cloud could reduce the execution and tardiness time and improve VHR utilization. The cost of using resources as an effective factor is also observed.

KEYWORDS: Expert Cloud, Load Balancing, Interval Estimation, Statistical Load Balancing Model.

1. INTRODUCTION

The technologies such as Cloud and Grid connect the physical world, human societies and information spaces together. HRs are among the important components of the organizations and societies. The organizations achieve success by the use of human knowledge, skills and experts. A platform called Expert Cloud presents the HRs which are geographically distributed. The knowledge, expert and skills of human beings are shared in this platform. These resources called Virtualized-HRs (VHRS) can utilize the provided specifications by other HRs such as discovering, ranking, knowledge all of which shared the Internet [1].

The cloud computing is a pattern based on distributed Internet. It is designed to make cooperation among the different resources and services through the network. It presents the powerful services according to the user's requirements. In other words, according to the definition of National Institute of Standards and Technology (NIST), cloud computing is the model to facilitate the easy access to the set of changeable processing resources and configuration through the network. This is done based on user request (such as networks, servers, applications, storages, etc.). This access should be able to be supplied or released with the minimum requirement to the resource management or direct service interference [2], [3].

Therefore, the existence of the heterogeneous dynamic VHRs from one hand and the variety of users request on the other hand cause the lack of load balancing in the environment. The load balancing distributes the resources in such a way no additional load occurs in any machine, and the resources are optimally utilized. However, it is not much paid attention to this aspect of cloud computing [4]. In the distributed computing systems such as parallel computing, Grid computing and also cloud computing, the load balancing algorithms are divided into static and dynamic categories according to the system topology and information resources availability as well. The static algorithms are not depended on the existing system state. These algorithms require prior knowledge from the system such as task requirements, the relationship time, system processing power, memory capacity, etc. Round Robin is a type of static algorithms. This is a simple method and uses fewer resources. It cannot detect the attacked server, and consequently it leads to unequal distribution [4]. These algorithms are not suitable for the distributed environments such as cloud. On the other hand, the dynamic algorithms such as GA are based on the existing system state and are used to confront the unpredictable load processing [5]. The load balancing scheduling algorithms can be divided into two groups: centralized and distributed [6], [7]. In the centralized method, a central controller is responsible for the load distribution, since this controller has a general view over the resources. It devotes each job to the appropriate resource. The problem of this method is bottleneck. The distributed scheduling algorithm makes the load balancing based on the dynamic information of the resources. The advantages of this method are the scalability and high fault tolerance.

In this paper, we present a new dynamic load balanced task scheduling algorithm in cloud computing. We present a mechanism for the improvement of some of the prior methods such as Round Robin, FIFO, Max-Min, etc. In spite of this fact that most of the algorithms have presented the solutions for the load balancing in the distributed environments, only a few number of algorithms have been proposed for the cloud environment. Most of the existing algorithms do not pay attention to the problems such as scalability, task specifications, the suitable mapping among tasks, resources, and the proportion between the allocation and service rate. The purpose of most of these algorithms is the task allocation to the idle resources without consideration the capability of these resources. They do not consider the proportion between the task arrival time and service rate as well.

Our proposed method attempts to improve execution time by solving these problems. The main idea of our method is that we model the load balancing and mapping process based on Simple Exponential Smoothing and Probability Theory. Our method uses the interval estimation and specification matrix to allocate the resources and also to determine the service rate. The process is in this way that this method uses GA as a ranking to make differences in VHRs capabilities. It utilizes the specification matrix to show the VHRs specifications. In continuation, the VHR service time is calculated by confidence interval and task mapping is done according to Poison process and Probability Theory. So the tasks are allocated to the VHRs which have the maximum execution capability and processing power. In addition to allocating, our method uses Exponential Smoothing to make load balancing. It improves the execution time and system performance. Simulation result proves the performance of our proposed method.

The rest of the paper is organized in such a way that we can introduce the related work in Section 2. In Section 3, we discuss the new method that we have presented. We express the simulation and the result of the proposed method in Section 4 and finally in Section 5 we conclude the paper.

2. RELATED WORKS

2.1. The load balancing is one of the important issues in heterogeneous computer networks. In fact, load balancing is the equal workloads distribution among the different computational resources. It leads to decreased response time, effective resource utilization, increased system performance, and elimination of the overload [8]. Different algorithms have presented different solutions that we intend to introduce them in this section.

To provide independent infrastructure services, the VMs has been created and developed. A virtual machine is a software implementation model of a real computer. It executes programs similarly to the main computer, but apart from that and with high performance. In fact, a virtual machine creates a complete system platform which supports the function of a full operating system. The VMs solve heterogeneous hardware and infrastructure problems. The same topics discussed about VMs are also about Human Resources (HRs) so that in the expert cloud, the expertise of HRs is provided as a service to others. In this infrastructure, the HRs become their virtual form. In this situation, the need for individuals to appear face to face disappears, and we can use their expertise as a virtual person without face-to-face interaction. Namely the main goal of the expert cloud is to share the expertise of HRs in the cloud platform as a Virtual Human Resource (VHR).

One of the algorithms has presented a technique named Live Virtual Machine [9]. In this method, the active VM is transferred from a virtual host to another. In continuation, it has proposed the task based load balancing by the use of swarm optimization in which the tasks migrate from overload VMs instead of overload VMs migration. The Cloudsim has been used to evaluate this method. The task scheduling model in this method is based on swarm optimization. This method decreases the execution time and energy [9]. One of the other algorithms has presented a mechanism of load balancing based on reservation policy to distribute the tasks among the replicated servers. In this method, overloaded servers store the capacity of the remote servers. Then, the requests are transferred to the remote ones. No server shares its capacity to the other one. In this way, the load transferring is limited among the servers. This method decreases the response time and the number of not responded answers [10]. Another algorithm utilizes the multi-core clustering processors to make load balancing. In order to select the best tasks for migration, this method uses the algorithms inspired of the nature and also to select the resources to accept these tasks. Each node selection is stochastic and based on previous knowledge of the node. This method increases the QoS and scalability [11]. Another load balancing algorithm is a dynamic method. This method is based on tree model and by inspiring of ants algorithms. This method puts the resources in the suitable sites; by calculating the competency rank. Tasks allocation is based on the tasks priority and control word. This method decreases makespan but it has high overhead [12]. Probabilistic modeling to achieve load balancing in Expert Clouds has been presented in [13]. This method provides the dynamic load balancing based on distributed queues aware of QoS in expert cloud. This method uses colorful ants to make distinction among resource capabilities. It provides the load balancing by labeling resources and based on Poison model. This method decreases makespan and tardiness [13]. Another method has proposed a scheduling method to achieve load balancing. This method consists of three algorithms. They are: on-demand scheduling, Querying and Migration Task (QMT) and also Staged Task Migration (STM). In a master-slave plan whenever the slave load is light, the master makes aware this slave of sending its extra loads. This step is done by QMT. In this method, QMT scheduling is suitable for dependent tasks and STM for independent ones. The disadvantage of this method is the high transferring time [14]. The presented method in [15] is a heuristic one which is based on sufferage value. It is to say that this method, primarily, calculates the subtraction of second earliest completion time and earliest completion time (sufferage value). Then, it allocates the tasks to the VMs according to this value. This method performs better than traditional ones such as Min-Min and Max-Min. One of the other existing methods which has been presented in [16] is LBMM (Load Balance Min-Min). This method is based on Opportunistic Load Balancing (OLB). OLB is a static algorithm and keeps the nodes busy without considering their execution time. This can slow down the processing time in turn [17]. LBMM adds three layers to the OLB architecture. The first layer is manager and it receives the tasks. The second layer is service manager; this layer divides each task into the subtasks to increase the processing speeds. Finally, this layer delivers the task to the third one [16]. Another static algorithm uses control scheduling controller and resource monitoring. The first one determines the resource for allocation and the second one scans the resources information. Then, the tasks are allocated to the resources with the highest score by using four phases which are: task submission, virtual machine request, receiving the resource information and resource capability computation [18]. Another method has addressed dynamic load balancing problem and solved it using Honey Bee Behavior inspired of Load Balancing (HBB-LB) algorithm. This method provides the load balancing among the VMs. In this algorithm, the tasks which should be balanced are regarded as honey bees and VMs as food sources. The bees destination is VMs with low load. This method is performed in four steps: 1. computing the current VM load, 2. making decisions about scheduling and load balancing. 3. VMs categorizing, and; finally, tasks scheduling. At the end, tasks (bees) are transferred from over loads VMs to underload ones. This method improves the QoS and decreases the makespan but its main problem is low scalability [19]. In [20], the MBA (market based control) method has been presented. This method provides the load balancing through data allocation and dynamic migration. In this method, the resources have been considered as traders in market. This algorithm performs data allocation and migration intelligently using market rules [21]. The load balancing process of this method is formed based on two agents: 1. Data- trading agent. 2. Data auctioning agent. The first one is responsible to follow the database nodes load and also the nodes price. The second one determines which traders can buy and which ones can sell. This method is dynamic and improves the response time. One of the other existing methods concerning resource management and load balancing is community based video replication. This algorithm clusters the social relationship with the near geographical situations that watch the similar videos. It presents this relationship in the form weighted graph, follower, and folllowee. In this method, the requests and tasks are scheduled based on community. The purpose of this method is to facilitate users access to set of CDN (Content Delivery Network) nodes which is in common with this community. This method decreases the cost and average service delay [22]. One of the methods of resource management and scheduling is Shuffled Frog Leaping Algorithm. The purpose of this algorithm is to model and mimic the behavior of a group of frogs to find food. They are randomly placed on a rock in a pond. This method is designed to schedule and manage the flow of various tasks in the infrastructure as a service, and it brings the benefits of genetic algorithm and particle swarm optimization. This algorithm, while minimizing the total cost of execution, ensures that tasks are performed within the deadline [23].

The existing algorithms in this section have presented the suitable methods for load balancing, but most of them aim to allocate the tasks to the idle resources. They have paid no attention to the resource capabilities, resource service rate, tasks and resource priorities and the probability of task execution in the resources. This can cause the task execution failure in the resources and increase response time and tardiness. To the best of knowledge, our proposed method is different from all the existing algorithms. It models the mapping and allocation process based on Simple Exponential Smoothing and Probability Theory. Our method presents the new solution by consideration of the existing algorithm problems. Contrary to the existing methods, our proposed method allocates the tasks by distinction among resource capabilities and also based on confidence interval and Poisson distribution. It performs the load balancing based on resource load prediction and specification matrix as well. Section 3 explains them in details.

3. PROPOSED METHOD: CONCEPTS

In spite of this fact that the existing algorithms in the field of load balancing have presented the suitable methods, most of them have paid no attention to the task priority and its specifications, resource capabilities, resource service rate, task execution probability over the resources and load prediction over them. Our proposed method presents the new solution by consideration of these problems. This new method utilizes the GA to make distinction among VHR capabilities. It uses the interval estimation (namely θ interval estimation is an interval in the form of $\widehat{\theta 1} < \theta < \widehat{\theta 2}$ in which $\widehat{\theta 1}$ and $\widehat{\theta 2}$ are the amount of the suitable random variables such as $\widehat{\theta 1}$ and $\widehat{\theta 2}$. The specified 1-α amount is a confidence interval for θ as well) to find the VHR service rate. In continuation, our method presents the specification of all VHRs in the form of specification matrix. It uses the Exponential Smoothing method to predict VHR load. Then, it performs mapping by calculation of the task execution probability in the VHRs. It is worthy saying that the tasks are separated according to their types. Finally, it controls the tasks arrival rate to the VHRs by forming allocation matrix and according to Poison Process. In the same way, the arrival rate becomes proportionate to service rate and the load balancing is achieved as well (Algorithm 1).

Algorithm 1 Our proposed algorithm at a glance.

```
1.  Begin
2.  Put VHRs in three sites based on GA
3.  For k=1 to z do // z is the number of VHRs in each site
4.  {
5.      $\Lambda_{VHRk} = \frac{1}{\eta}$
6.      $S_{VHRk} = \lambda^{-1}$
7.      $F_{VHRk} = F_{VHRi-1} + \beta\,(A_{VHRk-1} - F_{VHRk-1})$
8.      $P_{VHRij} = \frac{x_{ij}}{\sum_{k=1}^{m} x_{ik}} + \frac{E_{ij}}{\sum_{k=1}^{m} E_{ik}}$      $0 \le P_{ij} \le$ , $\sum_{i=1}^{n} p_{ij}=1$
9.   }
10. Forming the specification matrix for each site (m*4 matrix)
11. Get the task to the manager
12. For i=1 to n do // n is the number of tasks
13. {
14.     Sort the tasks based on types
15. }
16. Select the suitable site based on tasks priorities
17. For k=1 to z do // z is the number of VHRs in each site
18. {
19.     If number of input tasks > $\lambda_{VHRk}$ then
20.       $VHR_k$ is pruned
21.     Else
22.       {
23.         Select the candidate $VHRs_K$
24.         Check the $P_{VHRk}$ and $F_{VHRk}$ for candidate VHRs
25.         Select $VHRs_k$ with maximum $P_{VHRk}$ and minimum $F_{VHRk}$
26.          $S_{VHRk} = \lambda^{-1}$
27.         Allocate the tasks with $S_{VHRk}$ rate to the VHRs
28.       }
29. }
30. For k=1 to z do
31. {
32.   Forming the allocation matrix (m*n matrix)
33.    Put 1 value in each row of matrix after allocation
34.    Check $\sum_{i=1}^{n} T_i \le \lambda_{VHRk}$
35.    Do allocation until line 34 is established
36.  }
37. If some of input tasks cannot allocate do
38.     {
39. For k=1 to z do
40. {
41. $\sigma_k = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(F_{VHRi} - AF)^2}$
42. $AF_k = \sum_{i=1}^{m} \frac{F_{VHR}}{m}$
43.  Check $\sigma_k$ value
44. If $\sigma_k > f$ // f=[0-1]
45. {
46.   VHR is overload
47.   Send the tasks to the manager
48.    Go to line 19
49. }
50. If manager not find the VHRs in the first site
51.  {
52.     Go to site 2 and 3 based on tasks priority
53.      Repeat line 19 to 36
54.  }
55. end
```

3.1. The VHRS Ranking by GA

A genetic algorithm is a heuristic search that mimics the process of natural selection. This search (also sometimes called a meta-heuristic one) is routinely used to generate useful solutions to optimization and search problems [24]. We use the GA concept for making distinction among the VHR capabilities. In this method, each chromosome is considered as one VHR. If we consider $S = \{VHR_1, VHR_2, \ldots VHR_n\}$ as a set of chromosomes, we can choose the chromosome with the highest chance by the use of rank selection. In the rank-based selection scheme, chromosomes in the population are first stored according to their fitness values. Each chromosome's rank in the population is used to influence the selection process instead of its fitness value. Selection scheme is the following [24]:

$$Pi = \frac{R(i)}{\sum_{i=1}^{n} R(i)} \qquad i = 1,2,\ldots\ldots n \qquad (1)$$

Where, Pi denotes the selection probability of chromosome i, n denotes the population size and $R(i) \in [1,n]$ is the rank of chromosome i. In this case, $R(n)$ presents the best chromosome, and $R(1)$ is the worst one in the population. For the calculation of each chromosome rank, the fitness of each one should be calculated. In fact, the fitness of each chromosome depends on its average number of executed tasks by each VHR (λ_j) and average response time ($\bar{y}$).

$$Tfvi = \lambda j \quad i = 1,2,\ldots\ldots n \quad j = 1,2,\ldots.m \qquad (2)$$

$$STfvi = \bar{y} \qquad i = 1,2,\ldots\ldots n \qquad (3)$$

Where, Tfv_i and $STfvi$ denote the fitness from the view point of the number of the successful executed tasks and average response time respectively. It is worth noting that λ_j and $\bar{y}$ are calculated based on section (3.4). In continuation, Eq. (1) based on our proposed method is defined as follows:

$$PT = \frac{R_{Tfv(i)}}{\sum_{j=1}^{n} R_{Tfv\,(j)}} \qquad i=1,2,\ldots\ldots n \quad j = 1,2,\ldots.m \qquad (4)$$

$$PS = \frac{R_{STfv\,(i)}}{\sum_{j=1}^{n} R_{STfv\,(j)}} \qquad i = 1,2,\ldots\ldots n \quad j = 1,2,\ldots.m$$
(5)

Where, P_T is the indicator of the VHR selection probability based on a number of successful executed tasks, R_{Tfv} is the rank of each VHR from the view point of a number of successful executed tasks (λ_j value is high, the VHR rank is better), P_S is the VHR selection probability based on service time and finally R_{STfv} is the rank of each VHR from the view point of service time. In continuation, all VHRs are set in three sites based on P_T and P_S values. The VHRs with high P_T and P_S values are set in the first site, the VHRs with average P_T and P_S values are in the second, and finally the VHRs with low values are set in the third site. Eq. (6) shows this clearly.

$$Site\ Number = \begin{cases} 1 & if \quad A \le P_T \,\&\, P_S \le B \\ 2 & if \quad C \le P_T \,\&\, P_S \le A \\ 3 & if \quad D \le P_T \,\&\, P_S \le C \end{cases} \qquad (6)$$

In Eq. (6), A, B, C and D show the value of selection probability for each VHR. In our proposed method, the value of A, B, C and D are: 0.8, 1, 0.4 and 0 respectively. So, site 1 includes powerful VHRs (P_T and P_S with high values), site 2 includes average VHRs (P_T and P_S with average values) and finally site 3 includes weak VHRs (P_T and P_S with low values). Finally, VHRs are categorized by the use of GA concept.

3.2. Sites and Tasks Division based on Ranking

We use the VHR ranks to make sites. We consider three sites with tree structure for this purpose. The VHRs with high processing power are set in the first site, VHRs with the average processing power are set in the second site and; finally, the VHRs with low processing power are set in the third one. In order to determine the specifications of each VHR in sites, we allocate a label for it. Each label consists of three units. λ is the indicator of the number of executed tasks by each VHR, T_s is the VHR service time and β shows the rank of VHR. In addition to VHRs, the submitted tasks by the users should be labeled. Indeed, this label determines the type of user request. If the task should be immediately executed without delay; in this case, this task needs the powerful and fast VHR, the value of label will be one (tasks with high priority). If the task can wait for average time, then the value of task label will be two (tasks with average priority). At the last step, if a task can wait for a long time, the label value will be three (tasks with low priority).

3.3. Specification of Matrix Formulation

The specifications of VHRs in each site are displayed by a matrix. This matrix which is $m*4$ one, consists of m rows showing the VHRs and columns show the VHR specifications. These specifications are: λ which is the indicator of successful execution tasks by each VHR, S_{VHR} is the VHR service rate, P_{VHR} is the probability of task execution by each VHR and; finally, F_{VHR} shows the VHR load prediction. In the continuation, we will deal with these specifications (Eq. 7).

$$Sp = \begin{array}{c} \\ VHR_1 \\ VHR_2 \\ \vdots \\ VHR_m \end{array} \begin{array}{c} \begin{array}{cccc} \lambda & S_{VHR} & P_{VHR} & F_{VHR} \end{array} \\ \begin{bmatrix} sp_{11} & sp_{12} & sp_{13} & sp_{14} \\ sp_{21} & sp_{22} & sp_{23} & sp_{24} \\ \vdots & \vdots & \vdots & \vdots \\ sp_{m1} & sp_{m2} & \cdots & sp_{m4} \end{bmatrix} \end{array} \quad (7)$$

3.4. The Interval Estimation for Service Rate Calculation

One of the VHR specifications is service rate which is the same as task execution by each VHR. We use the interval estimation for the service rate calculation. We hypothesize that the *VHR = {VHR$_1$, VHR$_2$,….VHR$_m$}* are the VHRs set in each site in which each VHR can process n independent task: *T={T$_1$,T$_2$,.....T$_n$}*. If *t= {t$_1$,t$_2$,……t$_n$}* is task execution time by each VHR, then $f_\theta(t)$ will be probability distribution which depends on the unknown θ parameter. The purpose of this estimation is to find the quantity from the observed value *(t$_1$,t$_2$,……t$_n$)* as an approximation of unknown θ parameter. So by using the interval estimation, we calculate an interval time in which θ is to exist in this interval. It is hypothesized that two statistics θ_1 and θ_2 are such as follow:

$$\theta 1 = g_1(t) \quad (8)$$
$$\theta 2 = g_2(t) \quad (9)$$

We should calculate θ$_1$ and θ$_2$ in such a way that θ is to be placed in this interval if $P\{ \theta_1 < \theta < \theta_2 \}= 1\text{-}\alpha$ is to be established, then the purpose is to find the confidence interval. We can send the tasks to the VHRs with full confidence by using this value $(1\text{-}\alpha)$ according to (θ_1, θ_2) interval. So, we should calculate the estimated parameter for the calculation of the time interval which presents service to each VHR. For this purpose, we present the $(\bar{t} - a,\ \bar{t} + a)$ in which, with the $(1\text{-}\alpha)$ probability, the $E(\bar{t})$ will be inside this interval.

$$\bar{t}=\frac{1}{n}\sum_{i=1}^{n} ti \quad (10)$$

$$E(\bar{t})=\eta \quad (11)$$

$$\sigma_{\bar{t}}^2=\frac{\sigma^2}{n} \quad (12)$$

$\bar{t}$ is the average task execution time on each VHR and $E(\bar{t})$ is the expected value as well. σ^2 is also the variance for a set of *n* samples(tasks). After the calculation of these values, we can determine the interval time that each VHR can give services.

$$P\{\bar{t} - \frac{\sigma}{\sqrt{n}} * Z_{1-\frac{\alpha}{2}} < \eta < \bar{t} + \frac{\sigma}{\sqrt{n}} * Z_{1-\frac{\alpha}{2}} \} = 1\text{-}\alpha \quad (13)$$

$$Z=\frac{\bar{t}-\eta}{\frac{\sigma}{\sqrt{n}}} \quad (14)$$

Eq. (13) shows the confidence interval of each VHR in which Z is the normal standard parameter. It is calculated according to Eq. (14). So the service time of each VHR is inside θ$_1$ and θ$_2$ interval (namely (θ_1 , θ_2)).

$$S_{VHR} = (\theta 1 , \theta 2) \quad (15)$$

After the calculation of S_{VHR} , we can calculate the task interval rate to each VHR as well. It is to say that we want to calculate the number of successful tasks that each VHR can execute and accordingly allocate the task to each VHR. We use the Poisson distribution for this purpose.

$$\lambda = \frac{1}{E(\bar{t})} = \frac{1}{\eta} \qquad : \theta 1 < \eta < \theta 2 \quad (16)$$

$$\lambda = \frac{1}{S_{VHR}} = (\frac{1}{\lfloor\theta_1\rfloor}, \frac{1}{\lfloor\theta_2\rfloor}) \quad (17)$$

The Poisson distribution presents the number of successful executed tasks in an interval time. So we can calculate the λ value by using the S_{VHR} interval time which is the same as exponential distribution. Namely $S_{VHR}= \lambda^{-1}$. Thus, we can allocate the tasks with λ rate and in a confident interval time to each VHR by calculating the λ. In this way, all tasks are successfully executed. This means that we can make proportionate the interval rate with the service time.

3.5. Exponential Smoothing for VHR load Predection

Another specification of VHR is F$_{VHR}$ parameter. That is the same as VHR load prediction. The importance of this parameter is revealed during the allocation. It prevents from additional task allocation to some VHRs by being aware of the value of this parameter, and also it stops the lack of load balancing. We use the exponential smoothing to achieve this purpose. This method has a plenty of usage for the request prediction in the future. In this method, it is given different weights to different periods data. These weights follow a descending geometric progression. This method gives the maximum weight to the amount of the last period requests and whatever we go back; the weights decrease in the form of exponential. So, we use the Eq. (18) for the calculation of VHR load prediction and capacity.

$$F_{VHRi} = F_{VHR_{i-1}} + \beta(X) \qquad i = 1,2,3, \dots \dots m \quad (18)$$

$$X = A_{VHR_{i-1}} - F_{VHR_{i-1}}\ i = 1,2,3, \dots \dots m \quad (19)$$

In Eq. (18), F_{VHRi} is the indicator of VHR_i load prediction for the next period. $F_{VHR_{i-1}}$ shows the previous period VHR_i load. β is the smoothing coefficient and is acquired by trial and error. By consideration of some empirical information, the β value is between 0.1 to 0.3. $A_{VHR_{i-1}}$ is the indicator of the real VHR_i load request in the previous period. Finally, X shows the prediction error of previous period. So, we add this parameter as the third VHR specification to the specification matrix.

3.6. The Probability Theory

One of existing parameters in specification matrix is P_{VHRj}. That is the same as VHR probability selection to execute the tasks. We calculate the P_{VHRj} value for each VHR and choose the most probable VHR based on its value. It is hypothesized that $S=\{VHR_1, VHR_2, \ldots\ldots, VHR_m\}$ is non empty finite space. The probability model is presented as Eq. (20) over this sample space.

$$P_{VHR_{ij}} = \frac{x_{ij}}{\sum_{k=1}^{m} x_{ik}} + \frac{E_{ij}}{\sum_{k=1}^{m} E_{ik}} \; 0 \leq Pi \leq 1, \sum_{i=1}^{n} p_{ij} = 1, \quad j = 1,2,\ldots.m, = 1,2,\ldots n \tag{20}$$

$$x_{ij} = \frac{Task_i\ length}{VHR_j speed} \tag{21}$$

$$Eij = W_{VHR} \times (1 - L_{VHRj}) \times \frac{h_{speed}}{S} \tag{22}$$

$$L_{VHRj} = \frac{ETN}{TTN} \tag{23}$$

According to Eq. (20), $P_{VHR_{ij}}$ value contains performance evaluation of task execution over the VHR. Whatever the P_{ij} value is nearer to 1 , the VHR is more probable for the task execution. In this model, x_{ij} is the same as the task length ratio to each VHR speed. E_{ij} is calculated according to Eq. (22) in which W_{VHR} is the number of processors in each host and h_{speed} is the indicator of processor speed in each host. S is the minimum speed required for task execution as well. In this equation, L_{VHRj} is the same as the load of each VHR achieved from the division of number of present execution tasks into the number of total tasks (*TTN*). In this way, we can allocate the tasks to the VHRs which have the most chance of execution by calculating P_{ij} and consideration of its value. This, in turn, prevents the unsuccessful task execution.

3.7. Proposed Load Balancing Strategy

Our proposed load balancing is performed over four steps. This new method which is based on tree structure and mathematical model emphasizes on GA concept. The process is, in this way, that in the first step the existing VHRs based on GA are divided into three groups. In this way, the powerful VHRs (the resources with low service time and high number of successful execution tasks) are located in the first site; the VHRs with the average processing power in the second site, and, finally, the weak VHRs in the third one. After this step, the existing VHRs specifications in each site are displayed in the form of specification matrix. In this matrix, the λ, s, F and p columns are the indicators of the number of the successful executed tasks by each VHR; the service time (service rate) in each VHR, VHR load prediction and the probability of each task execution by VHR, respectively, this is the second step. In the third one, the user request enters the system. This request is transferred to the manager. The manager allocates the requests to the sites of one, two and three by considering the user type request. When the root node in each site which acts as a main parent, receives the task, then, it considers the specification matrix. The aim of this consideration is to prune the inefficient nodes in order to decrease the search space. Primarily, it compares the number of entered tasks with the λ column value in specification matrix and selects the candidate VHRs. The candidate VHRs are those which their λ value is greater or equal with the number of entered tasks. The parent node considers the second and third columns (F and P) after selecting the candidate nodes. It selects the node from the candidate ones in which the value of P_{ij} is higher (is nearer to 1) and also the F value (load prediction) is the least. In this step, the tasks are allocated to the related VHR based on existing service rate in s column. This is the same as task mapping model. It considers the service rate proportionate to arrival task rate based on Poisson model. After the allocation of tasks to each node, the allocation matrix is formed. This matrix is $n*m$; in which, the rows are tasks and the columns are VHRs. Each time that a task is allocated to each VHR, the value of its related element becomes 1 (Eq. 24). What is important in this matrix is the control of task allocation to each VHR.

$$AL = \begin{array}{c} \\ T_1 \\ T_2 \\ \vdots \\ T_n \end{array} \begin{array}{c} \begin{array}{ccccc} VHR_1 & VHR_2 & VHR_3 \cdots & & VHR_m \end{array} \\ \begin{pmatrix} AL_{11} & AL_{12} & AL_{13} & \cdots & AL_{1m} \\ AL_{21} & AL_{22} & AL_{23} & \cdots & AL_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ AL_{n1} & AL_{n2} & AL_{n3} & \cdots & AL_{nm} \end{pmatrix} \end{array} \tag{24}$$

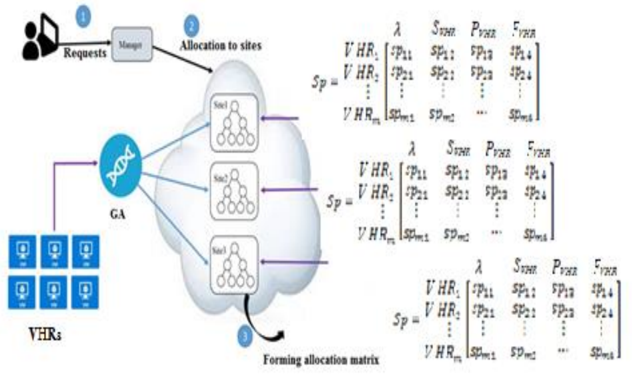
$$\sum_{i=1}^{n} T_i \leq \lambda_{VHRj} \quad i=1,2,\ldots\ldots n \quad j=1,2,\ldots..m \tag{25}$$

Eq. (25) means that the sum of the allocated task number to each VHR should be less than or equal the λ value of each VHR. In other words, the sum of each column values should be less than λ of each VHR. In this way, we can control the arrival rate to each resource, and allocate the number of tasks to each one which is capable to perform them. We allocate the tasks to each VHR by consideration of confidence interval (service rate) of each VHR and with full confidence in each S_{VHR} seconds as well. This means that we should proportion to the arrival rate with service time based on Poisson model. The fourth step is mentioned when the parent node can allocate just a number of received tasks to the one node (when a task processing in VHR is prolonged and the rest of tasks wait to get service). In this condition, the tasks which are not allocated should be migrated. In fact, the purpose of the fourth step is to keep load balancing. In this way that, the standard deviation value (Eq. (26)) is considered for each VHR in t_i to t_{i+1} interval. If σ value is in $k=[0-1]$ interval, the system is balanced. If $\sigma < k$, the system is underloaded, and finally, if $\sigma > k$, the system is overloaded [4]. So, the parent node transfers the additional load (not allocated tasks) to the undeloaded VHRs.

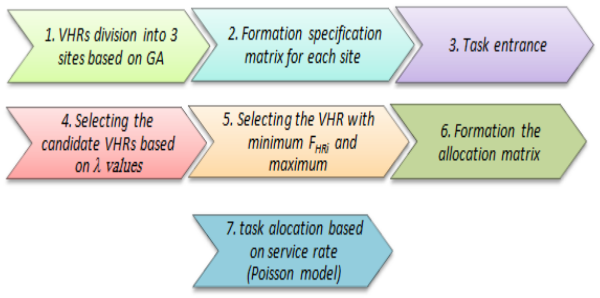
$$\sigma = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(F_{VHRj} - AF)^2} \tag{26}$$

$$AF = \sum_{j=1}^{m}\frac{F_{VHRj}}{m} \tag{27}$$

The transfer can be done in two forms: 1. the task migration should be in the related site. Namely, the parent node selects the VHR among the candidate ones according to the mapping rules by the consideration of P_{VHR} matrix that is to say its λ value is more equal than the number of entered tasks and P value is maximum while F one is minimum. It transfers the tasks in each s seconds. 2. The migration is not possible in the site. Namely, there is no node that has immigration conditions. In this condition, the tasks are transferred to other site. The priority of not allocated task migration in site 1 is primarily site 2 and, finally, is site 3. The priority of the task migration in site 2 is primarily site 1 and then site 3. Finally, the site 3 also considers; primarily, the site 1 to allocate the remaining tasks and then it goes to the site 2. In this way, the parent node in each site receives the not allocated task of another site, and allocates them based on mapping rules. So, the load balancing is kept in the sites as well.


1
Requests
Manager
2
Allocation to sites
Site1
Site2
Site3
GA
VHRs
3
Forming allocation matrix
$Sp = \begin{array}{c|cccc} & \lambda & S_{VHR} & P_{VHR} & F_{VHR} \\ VHR_1 & sp_{11} & sp_{12} & sp_{13} & sp_{14} \\ VHR_2 & sp_{21} & sp_{22} & sp_{23} & sp_{24} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ VHR_m & sp_{m1} & sp_{m2} & \cdots & sp_{m4} \end{array}$

(a)


1. VHRs division into 3 sites based on GA
2. Formation specification matrix for each site
3. Task entrance
4. Selecting the candidate VHRs based on λ values
5. Selecting the VHR with minimum F_{HRj} and maximum
6. Formation the allocation matrix
7. task alocation based on service rate (Poisson model)

(b)

Fig. 1. (a) The proposed environment model. (b) The proposed method workflow.

4. SIMULATION RESULTS AND ANALYSIS

In this section, we demonstrate the conclusion of the proposed algorithm with simulation. In this section, we have analyzed the performance of our algorithm based on the results of simulation using Cloudsim and validated it in Amazon EC2.We have expanded the Cloudsim classes for the simulation of our algorithm. The VHRs used to execute the tasks are modeled on a variety of Amazon M3 samples. We hypothesize that the number of tasks are 150. The flow of work and the length of tasks are also entered according to the standard dataset "LAG-2005.swf". It is supposed that the tasks are independently executed. The number of data center is one. It is hypothesized that the existing VHRs in the environment have been distributed in the form of 3 sites based on GA. Each site has tree structure; the maximum depth for each site is three. The existing VHRs in the environment are different because of their processing power. We consider the processing power for each VHR: 2, 1 and 0.5 seconds respectively. The evaluation parameters in this simulation are: makespan, tardiness, degree of load imbalance, speed up, and cost. Through careful attention to the mentioned quantities, makespan is computed as follow [4]:

$MakeSpan = max\{CTij \mid i \in T ,$
$i = 1,2,\dots,n \ and \ j \in VM \ , \ j = 1,2,\dots,m \}$
(28)

Eq. (28) shows makespan where, CT_{ij} is the indicator of $task_i$ completion time in VHR_j. Considering Fig. 2, axis x is the indicator of number of tasks and y axis shows makespan. The comparison between our proposed method and some of the other famous existing ones such as FCFS, RR, MET, Max- Min, Min-Min shows that our method has the lowest makespan. The maximum makespan in our method is 47720 s and minimum is 10850 s. This figure shows that FCFS has the most makespan (84330 s).

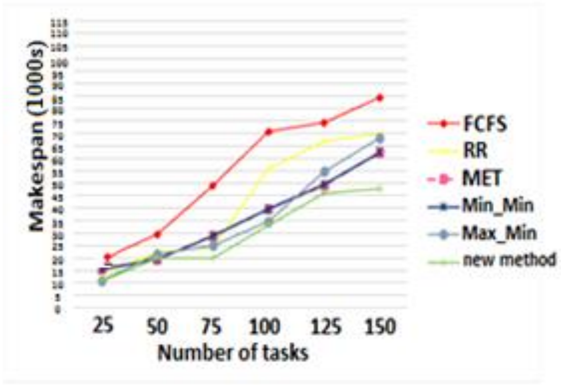


Fig. 2. Makespan comparison between methods.

Fig. 3 shows a comparison of the maximum completion time between the above methods as well. The effective reduction of the amount of makespan in the proposed method is due to the fact that, firstly, inputs are standardized in the simulation environment. Secondly, the proposed method addresses the effective distribution of tasks, which leads to a reduction in the execution time.

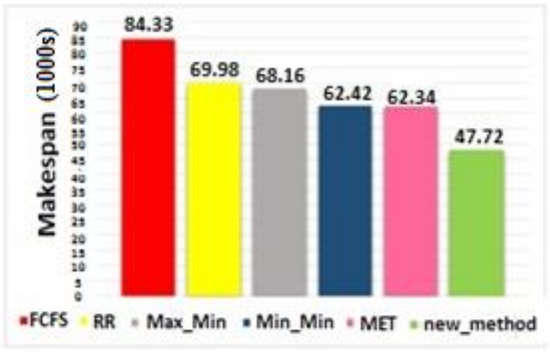


Fig. 3. Maximum makespan comparison between methods.

Afterwards, we calculate the tardiness. Tardiness is calculated according to the following equations.

$$Tardiness = finish\_time - dedline \qquad (29)$$

$$Dedline \ max_{time} + k \times (max_{time} - min_{time})$$

$$0 = k < 1 \qquad (30)$$

In the mentioned equations, *finish_time* is the end time and deadline is calculated based on Eq. (30). In this equation, *max_time* and *min_time* are highest and lowest execution times, respectively, and k is also between zero and one. After calculating the amount of tardiness and makespan for 150 tasks, Fig. 4 is obtained. This figure shows the relationship between makespan and tardiness. In this figure, axis x is the indicator of makespan and y axis shows tardiness. This figure shows that with the increase in the number of tasks and also the makespan, the amount of tardiness will also increase. This is due to the fact that with the increase in the number of tasks assigned to the system, the number of waiting tasks in the queue can be increased and finding VHRs appropriate to this number of tasks will require time. This could lead to delay. According to Fig. 4, the maximum tardiness is 55000s and the minimum is 25000s.

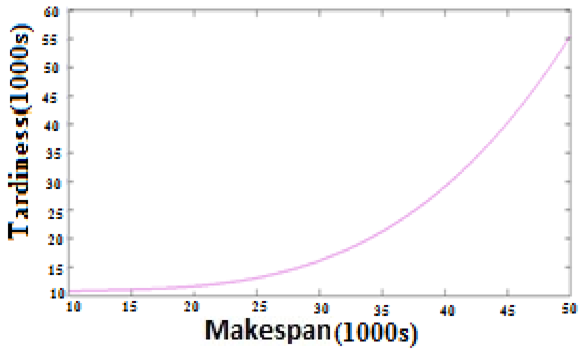


Fig. 4. Relationship between the Makespan and tardiness in our method

The next parameter to be calculated is *DI* (Degree of Imbalance). This parameter is calculated based on Eq. (31). Where *Tmax* and *Tmin* are the maximum and minimum *Ti* among all VHRs, *Tavg* is the average *Ti* of VHRs [4]. Our load balancing system reduces the degree of imbalance drastically. The calculation result of this parameter is shown in Fig. 5. In this figure, axis x is the indicator of number of tasks and y axis shows the *DI*. This figure shows that our proposed method reduces the *DI*. In this figure the maximum *DI* in proposed method is 14.3 and its minimum is 4.01.

$$DI = \frac{Tmax - Tmin}{Tavg} \qquad (31)$$

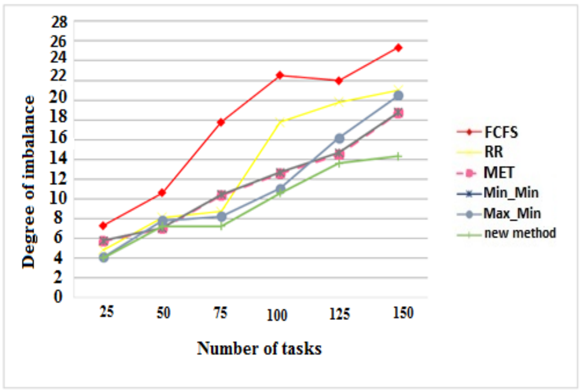


Fig. 5. DI comparison between methods.

The last parameter is the cost. The cost value is calculated according to Eq. (32). In this equation *Cost-Per-Sec* is the cost of execution a task on the resource. Based on this equation, Fig. 6 is obtained. In this figure, the x axis presents the number of tasks and the y axis shows cost. According to the obtained figure, the maximum cost is when the number of tasks is 150.

$$Cost = Cost_{Per_{Sec}} \times execution\_time \tag{32}$$

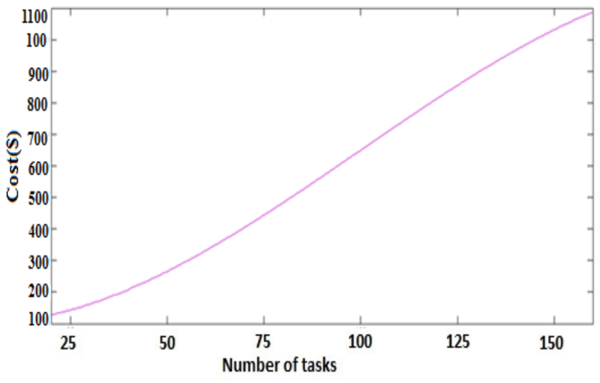


Fig. 6. Cost of proposed method

Fig. 7 also shows the cost comparison between the methods. In this figure, our proposed method has a maximum cost (1026.64 $), while the FCFS method has the lowest cost. The reason for the high cost in our proposed method is to check the quality of service before assignment. In that way, when the user requests the service, the system provides him with the appropriate resource for that request. This will prevent the failure of the implementation of work and assignment again. This means spending time and time.

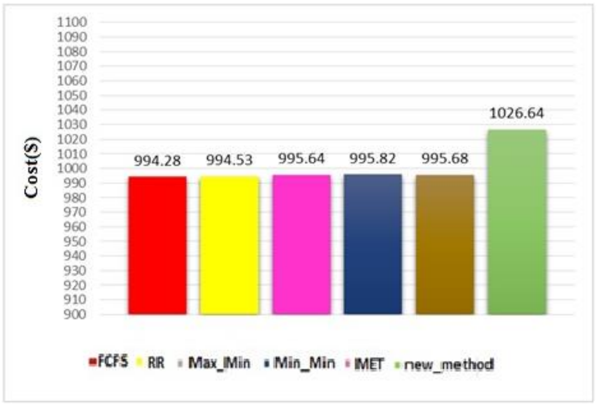


Fig. 7. Maximum cost comparison between methods.

5. CONCLUSION

In this paper, we have presented a new method for the dynamic load balancing in expert cloud. Our proposed method provides the load balancing and effective allocation by controlling the arrival rate and service time in each VHR. In this method, the GA has been used for making sites and distinction among the resource capabilities. The resource specifications have been expressed in the form of specification matrix. The tasks also have been categorized according to their types. The tasks allocation to the resources have been done according to Poisson model and probability theory. In our proposed method, the confidence interval has been used to estimate the service rate and based on this fact, the arrival rate has been calculated. In this way, it has presented the mapping model based on proportional arrival rate and service time. The proposed algorithm has provided the load balancing by controlling the arrival rate and also based on a mathematical prediction model. The proposed algorithm decreases the execution time and tardiness by effective allocation and making load balancing. We have compared the result of our proposed method with the traditional existing ones. The results show that our proposed method improves the execution time and cost in comparison with other existing ones.

For the future works, we intend to extend the load balancing for resources with more specifications such as fault tolerance and security. This method includes single point of failure problem which should be solved in future works. The use of other estimation methods such as variance estimation, likelihood and also the use of prediction methods such as linear regression could be used in the future researchers; it is proposed to develop the Cloudsim classes, as well.

REFERENCES

[1] N. Jafari Navimipour, A. Habibizad Navin, A. M. Rahmani, and M. Hosseinzadeh, **"Behavioral Modeling and Automated Verification of a Cloud-based Framework to Share the Knowledge and**

Skills of Human Resources", *Computers in Industry,* 15-16(42), pp. 6112–6131, 2015.

[2] P. Mell, and T. Grance, **"The NIST Definition of Cloud Computing"**, *National Institute of Standards and Technology*, pp. 1-7, 2011.

[3] F. H. Qusay,. **"Demystifying Cloud Computing"**, *the Journal of Defense Software Engineering*, pp. 16–21, 2011.

[4] D. Babu. L.D, and P.V. Krishna, **"Honey Bee Behavior Inspired Load Balancing of Tasks in Cloud Computing Environments",** *Applied Soft Computing*, 13(5), pp. 2292-2303, 2013.

[5] S. L. Chen, and Y.Y. Chen, **"CLB: A Novel Load Balancing Architecture and Algorithm for Cloud Services."** *Computers & Electrical Engineering* (58), 154-160, 2017.

[6] K.Q. Yan, S.C.Wang, C.P. Chang, and J.S. Lin , **"A Hybrid Load Balancing Policy Underlying Grid Computing Environment".** *Computer Standards & Interfaces*, Vol. 29(2), pp. 161-173, 2007.

[7] J. Balasangameshwara, and N. Raju**, "A hybrid policy for Fault Tolerant Load Balancing in Grid Computing Environments"**. *Journal of Network and Computer Applications*, Vol. 35(1), pp. 412-422, 2012.

[8] P. Kuila, and P. K. Jana,. **"Approximation Schemes for Load Balanced Clustering in Wireless Sensor Networks",** *The Journal of Supercomputing*, Springer (68), pp. 87-105, 2014.

[9] F. Ramezani, j. Lu, and F.K. Hussain, **"Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization."** *International Journal of Parallel Programming*, Vol. 42(5), pp. 739-754, 2013.

[10] A. Nakai, E. Madeira, and L.E. Buzato,**"On the Use of Resource Reservation for Web Services Load Balancing."** *Journal of Network and Systems Managemen,t*, Vol. 23(3), pp. 502-538, 2014.

[11] A. De Falco, E. Laskowski, R. Olejnik, U. Scafuri, E. Tarantino, and M. Tudruj, **"Extremal Optimization Applied to Load Balancing in Execution of Distributed Programs."** *Applied Soft Computing*, Vol. 30, pp. 501-513, 2015.

[12] L.M Khanli, S. Razzaghzadeh, and S.V. Zargari, **"A New Step Toward Load Balancing Based on Competency Rank and Transitional Phases in Grid Networks"**, *Future Generation Computer Systems*, Elsevier, Vol. 28, pp. 682-688, 2012.

[13] S. Razzaghzadeh, A.H. Navin, A.M. Rahmani, and M. hosseinzadeh , **" Probabilistic Modeling to Achieve Load Balancing in Expert Clouds"** , *Ad Hoc Networks*, Elsevier, Vol. 28, pp.12-23, 2017.

[14] Y. Liu, C. Zhang, B. Li, and J. Niu**, "DeMS: A Hybrid Scheme of Task Scheduling and Load Balancing in Computing Clusters."** *Journal of Network and Computer Applications,* Vol. 83, pp. 213-220, 2015.

[15] Maheswaran, M., et al. **"Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems",** *Journal of Parallel and Distributed Computing*, Vol. 59: 107-131, 1999.

[16] S.C. Wang, K.Q. Yan, W.P. Liao, and S.S. Wang, **"Towards a Load Balancing in a Three-Level Cloud Computing Network"**, *3rd International Conference on Computer Science and Information Technology (ICCSIT)*, IEEE, 2010.

[17] A. Sang, X. Wang, M. Madihian, and R.D. Gitlin, **"Coordinated Load Balancing, Handoff/Cell-Site Selection, and Scheduling in Multi-Cell Packet Data Systems"**, *Wireless Networks*, Vol. 14(1), pp. 103–120 , 2008.

[18] J. Ni , Y. Huang, Z. Luan, J. Zhang, and D. Qian, **"Virtual machine mapping policy based on Load Balancing in Private Cloud Environment"**, *International Conference on Cloud and Service Computing (CSC),* IEEE, pp. 292–295, 2011.

[19] K.R. Babu, and P. Samuel, **"Enhanced Bee Colony Algorithm for Efficient Load Balancing and Scheduling in Cloud"**. *Innovations in Bio-Inspired Computing and Applications: Proceedings of the 6th International Conference on Innovations in Bio-Inspired Computing and Applications (IBICA 2015)* held in Kochi, India during December 16-18, 2015. V. Snášel, A. Abraham, P. Krömer, M. Pant and K. A. Muda. Cham, Springer International Publishing**:** pp. 67-78, 2016.

[20] T. Wang, Z. Lin, B. Yang, J. Gao, A. Huang, D. Yang, Q. Zhang, S. Tang, and J. Niu, **"MBA: A Market-Based Approach to Data Allocation and Dynamic Migration for Cloud Database,"** *Science China Information Sciences*, Vol. 55(9), pp. 1935-1948, 2012.

[21] J. Niu, K. Cai, E.H. Gerding, and S. Parsons, **"Characterizing Effective Auction Mechanisms: Insights from the 2007 TAC Market Design Competition"**. *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems* - Volume 2. Estoril, Portugal, International Foundation for Autonomous Agents and Multiagent Systems**:** pp**.** 1079-1086, 2018.

[22] H. Hu, Y. Wen, T.S. Chua, J. Huang, W. Zhu, and X. Li, **"Joint Content Replication and Request Routing for Social Video Distribution Over Cloud CDN: a community clustering method",** *IEEE Trans. Cir-cuits Syst. Video Technol.* Vol. 26(7), pp. 1320–1333 , 2016.

[23] P. Kaur, and S.Mehta, **"Resource Provisioning and Work Flow Scheduling in Clouds using Augmented Shuffled Frog Leaping Algorithm"**, *Journal of Parallel and Distributed Computing*, Vol. 101, pp. 41-50, 2017.

[24] M. Mitchell, "**An Introduction to Genetic Algorithms"**. *Cambridge, MA: MIT Press.* ISBN 9780585030944.