I. J. Computer Network and Information Security, **2024, 2, 65-78**
Published Online on April 8, 2024 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2024.02.06

Individual Updating Strategies-based Elephant Herding Optimization Algorithm for Effective Load Balancing in Cloud Environments

Syed Muqthadar Ali*
Department of Computer Science and Engineering, Annamalai University, Annamalainagar, Chidambaram 608002, Tamil Nadu, India
E-mail: smuqthadarali34@gmail.com
ORCID iD: https://orcid.org/0000-0001-7645-8577
*Corresponding Author

N. Kumaran
Department of Computer Science and Engineering, Annamalai University, Annamalainagar, Chidambaram 608002, Tamil Nadu, India
E-mail: kumaran81@gmail.com
ORCID iD: https://orcid.org/0009-0000-4536-2775

G.N. Balaji
School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India
E-mail: balaji.gnb@gmail.com
ORCID iD: https://orcid.org/0000-0002-5346-0989

Received: 01 March 2023; Revised: 25 May 2023; Accepted: 22 August 2023; Published: 08 April 2024

Abstract: In this manuscript, an Individual Updating Strategies-based Elephant Herding Optimization Algorithm are proposed to facilitate the effective load balancing (LB) process in cloud computing. Primary goal of proposed Individual Updating Strategies-based Elephant Herding Optimization Algorithm focus on issuing the workloads pertaining to network links by the purpose of preventing over-utilization and under-utilization of the resources. Here, NIUS-EHOA-LB-CE is proposed to exploit the merits of traditional Elephant Herd Optimization algorithm to achieve superior results in all dimensions of cloud computing. In this NIUS-EHOA-LB-CE achieves the allocation of Virtual Machines for the incoming tasks of cloud, when the number of currently processing tasks of a specific VM is less than the cumulative number of tasks. Also, it attains potential load balancing process differences with the help of each individual virtual machine's processing time and the mean processing time (MPT) incurred by complete virtual machine. Efficacy of the proposed technique activates the Cloudsim platform. Experimental results of the proposed method shows lower Mean Response time 11.6%, 18.4%, 20.34%and 28.1%, lower Mean Execution Time 78.2%, 65.4%, 40.32% and 52.6% compared with existing methods, like Improved Artificial Bee Colony utilizing Monarchy Butterfly Optimization approach for Load Balancing in Cloud Environments (IABC-MBOA-LB-CE), An improved Hybrid Fuzzy-Ant Colony Algorithm Applied to Load Balancing in Cloud Computing Environment (FACOA-LB-CE), Hybrid firefly and Improved Multi-Objective Particle Swarm Optimization for energy efficient LB in Cloud environments (FF-IMOPSO-LB-CE) and A hybrid gray wolf optimization and Particle Swarm Optimization algorithm for load balancing in cloud computing environment (GWO-PSO-LB-CE).

Index Terms: Elephant Herding Optimization Algorithm, Load Balancing, Cloud Computing.

1. Introduction

The technology of cloud computing (CC) is computed to express the phenomenal requirement in the network technology field due to advancement in communication tools, capability to solve large scale issues, explosive internet application. It allows software as well as hardware applications as resources for cloud user across the Internet [1]. Furthermore, CC is one of the computing models that utilizes internet to share the resources, like software, servers, applications, services, storage [2]. The measurable with proficient behaviors of CC is feasibly persistent by the effectual maintenance of cloud resources. The virtualization of cloud resources is a significant feature of cloud computing [3]. In a rented basis, the service to the users is offered through the cloud service provider (CSP). CSP provide service to the users is deemed as a complex problem with available virtual cloud resources. Therefore, researchers focus more in the features of load balancing [4, 5]. Establishing the LB process between the virtual machine/physical hosts [6,7]. The basic benefit of LB is focused on workload balance within the host of cloud environment depends upon their capacity, which is assessed by bandwidth, obtainable speed of memory and processor [8, 9].

It can be challenging to manage incoming user requests and tasks then maintain a balanced workload in cloud systems owing to incorrect Virtual Machine (VM) allocation. It is just a few task characteristics are considered [10, 11]. For eg, cloud systems where the requests are not prioritized, it is impossible for all tasks to arrive at once if the arrival time is not considered. Performance issues could occur as a result of the increasing number of requests due to an uneven load in the cloud, particularly if the requests do not assign to the suitable VM or CPU is not fully utilized or unable to handle the demands [12, 13]. An efficient approach that improves cloud performance for IaaS while taking QoS into account must be offered to address these issues [14, 15]. It is made by optimizing the utilization of system resources that lessens the Makespan and Execution time of user tasks [16]. The authors focused their research's limited scope on improving task scheduling and load balancing in the cloud. This is the main motivation of this work. The main goal of this work is loading balancing in the cloud computing. Increased cloud usage by clients could result in incorrect job scheduling inside the system [17, 18]. As a result, problems with task scheduling should be solved using an Elephant Herd Optimization algorithm algorithm, which will be discussed in more detail in proposed section.

This manuscript consist of an NIUS-EHOA-LB-CE is proposed to facilitate an effective load balancing process in cloud computing. This NIUS-EHOA-LB-CE is proposed to exploit the merits of traditional Elephant Herd Optimization (EHO) algorithm in order to achieve superior results in all dimensions of cloud computing. This proposed NIUS-EHOA-LB-CE prevented the shortcomings of the existing metaheuristic algorithms in attaining superior load balance between the physical machines.

The key contribution of this manuscript is abridged below,

- NIUS-EHOA is proposed for facilitating the effective load balancing process in cloud computing (NIUS-EHOA-LB-CE).
- The primary goal of the proposed NIUS-EHOA [19] is to distribute the workloads pertaining network links for the purpose of preventing over-utilization and under-utilization of the resources.
- The proposed NIUS-EHOA-LB-CE exploiting the merits of the Elephant Herd Optimization (EHO) algorithm to achieve superior results in all dimensions of cloud computing.
- In this, NIUS-EHOA-LB-CE achieves the allocation of Virtual Machines (VMs) to the incoming tasks of cloud, when the number of currently processing tasks of a specific VM is reduced than cumulative count of tasks presently processing the other virtual machines in the cloud.
- It also attains potential load balancing process, then the variation among the processing time of every individual virtual machine and the mean processing time (MPT) incurred by the complete virtual machine. Finally, the simulation of NIUS-EHOA-LB-CE is conducted utilizing Cloudsim platform.
- The simulation results highlighted the efficiency of NIUS-EHOA-LB-CE system that is to be evaluated based on MRT under various number of tasks, MRT under various executable instruction lengths, Mean Execution Time under different numbers of task, count of migrated tasks with different count of virtual machine is analyzed.
- Finally, the efficacy of proposed approach is analyzed with existing IABC-MBOA-LB-CE [20], FACOA-LB-CE [21], FF-IMOPSO-LB-CE [22] and GWO-PSO-LB-CE [23] models.

Continual manuscript is organized as: section 2 reviews the literature review of different research papers related to efficient Load Balancing of Cloud Environments, section 3 explains the proposed technique, section 4 proves the results and section 5 concludes this manuscript.

2. Literature Review

More research works were previously presented in the literature associated to effective LB process in cloud computing. Among them, a few works are reviewed here,

Janakiraman and Priya, [20] have presented the Improved Artificial Bee Colony utilizing Monarchy Butterfly Optimization approach for LB in cloud computing for efficient resource usage in clouds and it comprises global exploration capacity of Artificial Bee Colony, local exploitation potential of Monarchy Butterfly Optimization Algorithm for allocating user tasks to virtual machine. Network focus and computing resources prevents the fragmentation, increase unnecessary task and the finishing times be explored potentially for improving the resource allocation procedure.

Ragmani et al., [21] have presented the CC required for effectual using the presence of static and dynamic scheduling strategies to enhance the grade of user fulfillment. Where the LB methods consider the possibilities of every individual virtual machine, executable instruction length on multiple task reliance was necessary for dynamic scheduling arriving tasks. This presented strategy considered the load balancing procedure an NP issue including metaheuristic models

Devaraj et al., [22] have presented the energy efficient LB in cloud computing utilizing FIMPSO algorithm. The presented algorithm attained actual total load for making and improved the necessary measures of resource usage like proper, response time tasks. The simulation outcome shows that the measures were utilized to evaluate like resource utilization, processing time, life span, reliability, throughput.

Gohil and Patel, [23] have introduced hybrid PSO-GWO algorithm for load balancing in CC environment. PSO-GWO was utilized based on the heterogeneous resources in LB that were likened to other models. The simulation outcomes proofs that the degree of unbalance measure as well as various counts of iterations, search agents and run.

Jena et al., [24] have suggested Q learning process for load balancing improvement in cloud environment. the Q learning with MPSO was utilized to develop the evaluation metrics along coverage rate for LB. Where, load for every virtual machine and balance it via the minimization of fitness function which was determined by Q learning based MPSO approach. The suggested method was to enhance the Q learning saves the Q values of better achievement of the stage, thus saves the storage space and velocity.

Shafiq et al., [25] have introduced the LB cloud computing that improves the distribution of workload, resource proficiency usage for reducing the response time in general system. For solving the load balancing problems in cloud environment, such as migration, scheduling of tasks, and utilization of resource. Where, presented a vital challenge in cloud computing and the problems associated with load balancing. The study was to identity the research problems based on load balancing decreases the response time, also it avoids the server failure in the existing load balancing method.

Joshi et al., [26] have suggested the load balancing technique in cloud computing environment to effectively utilize the compute resources. Cloud computing service obtainability was a projecting requisite. But occasionally it was difficult for meeting the service provider in inevitable reasons, viz unavailable power, temperatures, unavailable internet. In these cases, suggested approach was helpful for service consumers for unstable the tasks for harmless environment from difficult cloud environment.

Polepally and Shahu Chatrapati, [27] have suggested the LB algorithm based upon constraint measure. Tasks were allotted to virtual machine under round robin scheduling algorithm in load and capability of all virtual machine was calculated. Suppose the balancing threshold value was higher than virtual machine load, then the load balancing algorithm utilizes the task scheduling over the VM. In decisive factors were measured using load balancing algorithm to every virtual machine and fills the decisive list. Performances for the suggested technique were compared with existing method.

3. Proposed Methodology

Here, an individual updating schemes based elephant herding optimization algorithm are proposed to facilitate the effective load balancing process in cloud computing (NIUS-EHOA-LB-CE). The block diagram of proposed NIUS-EHOA-LB-CE method is given in Fig. 1. The proposed methodology figure shows the requests from a variety of clients (application users), including both desktop and mobile users. To make requests to the cloud, clients can use a variety of devices to connect to the Internet. The method uses the Cloudlet Scheduler Time Shared method in this layer to submit tasks at random sequence (arrival time) and schedule them to VM while taking 2 key factors into consideration: the deadline and the completion time. Data Centres (DC) in cloud computing are large repositories for cloud servers with data. Requests are sent to active load balancer by DC after it receives them. It has a main batch of VM, since it doesn't comply with the SLA, its status has been changed to high priority, meaning it will be completed before the deadline. By altering the MIPS of VMs prior and after providing the resources to them, the proposed LBA use a migration strategy to transfer the burden to another available Virtual Machine. The quantities of requests allotted to each Virtual Machine and if they have been violated are then updated in the allocation table. In one instance, there was no SLA breach. Let's say that for jobs to run on VMs, the Time to Complete is shorter than the SLA provided. After that, no SLA violation happens. The detailed discussion regarding an individual updating schemes based elephant herding optimization algorithm that facilitate the effective load balancing process of cloud computing are given below,

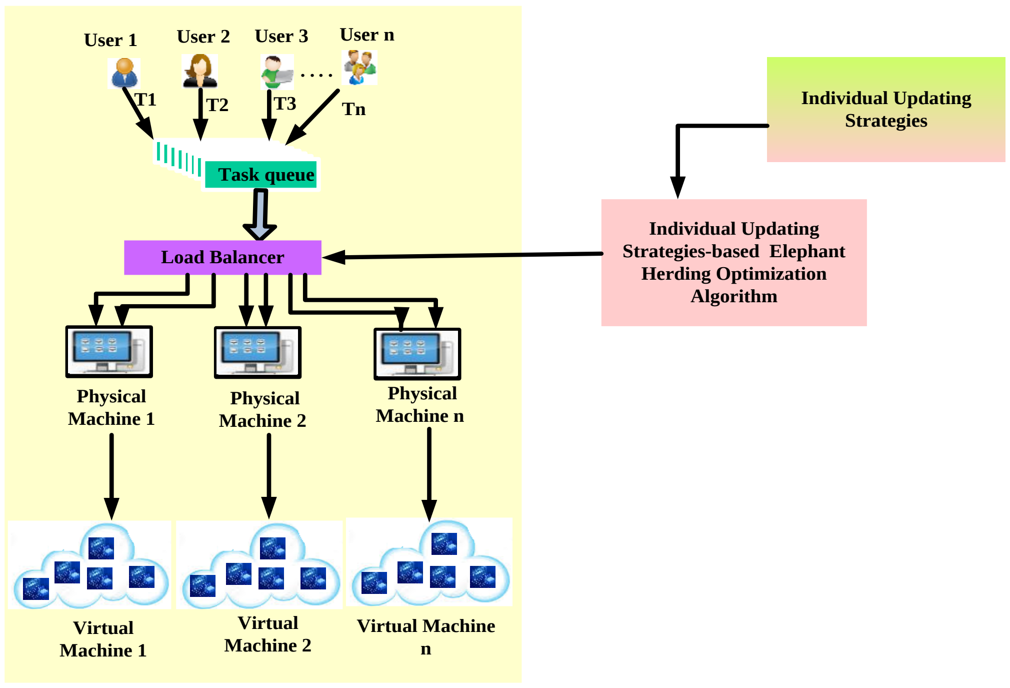

Fig.1. Block diagram of proposed NIUS-EHOA-LB-CE method

3.1. Implementing the Individual Updating Strategies-based Elephant Herding Optimization Algorithm (NIUS-EHOA-LB-CE)

In this section, description and methodical steps involved to execute the proposed NIUS-EHOA-LB-CE system. The proposed NIUS-EHOA-LB-CE scheme consist of the cloud computing environment comprises of set of virtual machines with associated tasks for processing. The complete sets of VMs are considered to be parallel and unrelated with non-pre-emptive independent tasks scheduled to them. In other works, the processing of task by a virtual machine cannot be interrupted. Thus, the proposed model assumes that the failure does not happen. This cloud computing environment is considered to comprise of a collection of data centers, the data centers in turn consists of hosts and each host includes a collection of k virtual machine. The individual data center consists of virtual machine load balancer. This load balancer is responsible for identifying an appropriate host and suitable VM from the selected host for the objective of allocating the subsequent task by computing some specific metrics. The metrics used for task allocation includes, processing time of host, MPT of every host, processing time of VM, and MPT of every virtual machine, load standard deviation, standard normal deviation of VM, and available virtual machine for the computation functions.

The running time of host is computed as a ratio of overall length tasks submit with particular host to the product of count of processors and the running speed of host. Otherwise, it is defined as the ratio of average request number length to the particular host to product number of processors, processing speed of host. Then, the host processing time is determined based on equation (1) as follows,

$$Host_{PT}(s) = \frac{Host_{LT}(s)}{Host_{N(P)}(s)+Host_{PS}(s)} = \frac{\sum_{s=1}^{k} Rq_{Length}(s)}{Host_{N(P)}(i)+Host_{PS}(s)} \tag{1}$$

where average length of tasks submitted to the particular host that is represented as $Host_{LT}(s)$, number of processors are expressed as $Host_{N(P)}$, processing speed of host is represented as $Host_{PS}$ and total request number length is expressed as Rq_{Length}. The MPT of the whole set of hosts is defined as total running time of host is divided by a number of hosts in the cloud platform. The MPT of all hosts are determined by equation (2) as follows,

$$Host_{M(PT)}(s) = \frac{\sum_{s=1}^{k} Host_{PT}(s)}{k} \tag{2}$$

where k specifies number of hosts in cloud platform. Processing the time of VM is computed based on the ratio of average length task, that is submitted for particular virtual machine with the product of count of processors and the processing speed of virtual machine. Otherwise, ratio of total request number length to the particular VM along with the product of the count of processors and processing speed of virtual machine. Then, the processing time of virtual machine is given below in equation (3)

$$VM_{PT}(s) = \frac{VM_{LT}(s)}{VM_{N(P)}(s)+VM_{PS}(s)} = \frac{\sum_{s=1}^{k} Rq_{Length}(s)}{VM_{N(P)}(s)+VM_{PS}(s)} \tag{3}$$

where $VM_{LT}(s)$ represents the average length of tasks that is submitted to the particular virtual machine, number of processors are represented as $VM_{N(P)}$, processing speed of the VMs are denoted as VM_{PS} and total request number length is represented as Rq_{Length}. MPT of virtual machine entire set is determined as sum of running time virtual machine is divided using the count of VMs in cloud platform. It is determined by equation (4) as follows,

$$VM_{M(PT)}(s) = \frac{\sum_{s=1}^{k} VM_{PT}(s)}{N(M)} \tag{4}$$

Let $N(M)$ represents number of VM in cloud platform. The standard deviation of load in cloud computing are computed based on the dispersion or variation existing with process time of virtual machine, the mean process of virtual machine to number of virtual machines in cloud platform also it is determined by equation (5) as follows,

$$SD_{Load} = \sqrt{\frac{\sum_{s=1}^{k}(VM_{PT}(s)-VM_{M(PT)})^2}{N(M)}} \tag{5}$$

where VM_{PT} represents the processing time of virtual machine and the mean process virtual machine are represented as $VM_{M(PT)}$. The standard normal deviation of virtual machine is specified as the ratio of processing time for every virtual machine, MPT of complete virtual machine in cloud platform with load standard deviation. It is computed by equation (6) as follows,

$$VM_{SND}(s) = \frac{(VM_{PT}(s)-VM_{M(PT)})}{SD_{Load}} \tag{6}$$

The availability of virtual machine is identified based on deviation of processing virtual machine from MPT is equivalent with entire set of virtual machines that is less or equivalent to threshold limit.

3.2. Description of the Individual Updating Strategies-based Elephant Herding Optimization Algorithm (NIUS-EHOA-LB-CE) Scheme

The proposed method makes effectual LB process in cloud computing with maximizing throughput objective in CC deals through balance the task priorities above the virtual machine i.e., waiting time is less with count of task is presented in queue. The proposed method prevents the over loading and under loading virtual machine that guarantees the substantial response time, optimal machine deployment that is recognized for reducing the waiting time, MET of tasks which are available in the queue. This system offers an efficient allocation of incoming tasks to virtual machine that satisfies the 2 important conditions. Condition one shows the numbers of task which is implemented particularly virtual machine needs to be less than the average number of tasks that is presently implemented using the remaining virtual machine. Condition two focus on the fact, difference between processing the specific virtual machine then the MPT of the whole set virtual machine is lesser than threshold value. By using the individual updating approaches in Elephant Herding Optimization into load balancing phase verifies the available virtual machine, hosts that are deliberated as follows.

Initially, the numbers of hosts are represented as $(s = 1,2,...k)$ and the VMs are expressed as $(t = 1,2,....,m)$ represents swarm. Initially the host and virtual machine is selected to produce the uniform distribution, that is indicated in equation (7),

$$y_q^s = y_q^{\min} + RN[0,1] * [y_q^{\max} - y_q^{\min}] \tag{7}$$

where s and t specifies the presently searched hosts/virtual machine, an arbitrarily selected host among the average number of hosts. s represents the host/virtual machine which could be allocate for tasks with $y_q^{\max}$ and $y_q^{\min}$ represents the limit of maximum and minimum y^s in the q^{th} dimensions and $RN[0,1]$ represents random count with uniform distribution between 0, 1 range. The search for allocating hosts and virtual machine is adjusted based on the below equation (8),

$$V_q^s = y_q^s + \phi_q^s(y_k^s - y_k^q) \tag{8}$$

where S is the currently searched host or VM with k as the host selected/virtual machine among the total number of hosts/VM like $k \neq s$; ϕ_q^s represents the random number that is between the interval [-1, 1] with q as the dimensions are chosen for verifying the availability of hosts , virtual machine to tasks. It is clear that the solution of modernizing is influenced using the information that is derived from y_k^q solution. This consequent solution affects the qualities of upgraded solution, also it is necessary to maintain the divergence in whole search space. Therefore, equation (8) is altered because it manages the diversity degree, rather it completely relies on selecting the solutions y_k^q. Modified virtual machine position are expressed below equation (9),

$$V_q^s = y_q^s + \phi_q^s(y_k^s - y_s^k) + \psi(x_q - y_q^s) \tag{9}$$

where x_q relates with the q^{th} dimension of potential virtual machine/host is measured for allocating the tasks of incoming to cloud computing, ψ represents the random number which is presented in-between 0 and with the positive constant that is allocated by the user. Next stage is to initiate the exploitation process in search process to identify the optimal hosts/virtual machine that are chosen for tasks that is assigned by suitable probability for selecting the hosts/virtual machine from the neighborhood. Thus, the new neighboring virtual machine/hosts are determined for new allocations for using the elephant herding optimization algorithm that process the optimal convergence speed with primitive individual updating strategies. Additionally, the individual updating strategies is responsible to exploit the present availability of hosts/ virtual machine to effectively incorporating the adjusting parameter of elephant herding optimization algorithm. Hence, to improve the searching ability and avoid the local optimum, here, individual updating approaches is with elephant herding optimization algorithm.

3.3. Step by Step Procedure for NIUS-EHOA

Here, in the cloud computing scenario, elephant herd optimization algorithm (EHOA) is proposed to attain suitable balance of virtual machine along with the objective of maximizing the throughput. It handles with balancing the task priorities across the virtual machine like reducing the waiting time is based on count of tasks is presented in queue. This method is proposed to prevent over loading virtual machine and under loading virtual machine for guarantee the substantial response time, optimal machine application. This is recognized for decreasing the waiting time, the task of MET is presented in the queue. The Individual Updating Strategies based NIUS-EHOA is proposed to distribute the workloads that are related with multiple network links to prevent under and over the resource of utilization. Fig. 2 shows the flowchart of NIUS-EHOA. Moreover, Fig. 2 depicts the flowchart of the steps included during the NIUS-EHOA. NIUS-EHOA is utilized for LB in cloud computing environment with the help of fitness function. The stepwise process of NIUS-EHOA is given beneath,

Step 1: Initialization

Randomly initialize the population p of NP elephant individuals along with uniform distribution of search space. Then, the set numbers kept the elephant as KEL_n, Max_{Gen} represents the maximal generation, p and βrepresents scale factor, p denotes count of clan, the count of elephants for ${D_i}^{th}$ clan is n_{di}.

Step 2: Random generation

During initialization process, randomly create the input parameters. Here, the maximum fitness values are designated dependent on exact hyper parameter condition. In this, the population of process value is created randomly for increasing the throughput and in the cloud computing scenario.

Step 3: Fitness Function

This is evaluated for reducing the execution time of data center to attain the objective function, the whole response time disturbs the task workloads through distinct virtual machines Fitness probability is utilized to select the optimal host/virtual machine from neighborhood that is related to equation (10)

$$Pr_{fitness}(R) = \frac{0.9 * fitness_i}{fitmess\_max} \tag{10}$$

where $fitness_i$ represents fitness of virtual machine/hosts called allocation. The fitness function is equivalent to objective function's negative value in case of minimization issue that in context with maximizing the optimization problem. $fitmess\_max$ denotes value of fitness, superior solution (virtual machine/host) is determined up to the previous iterations.

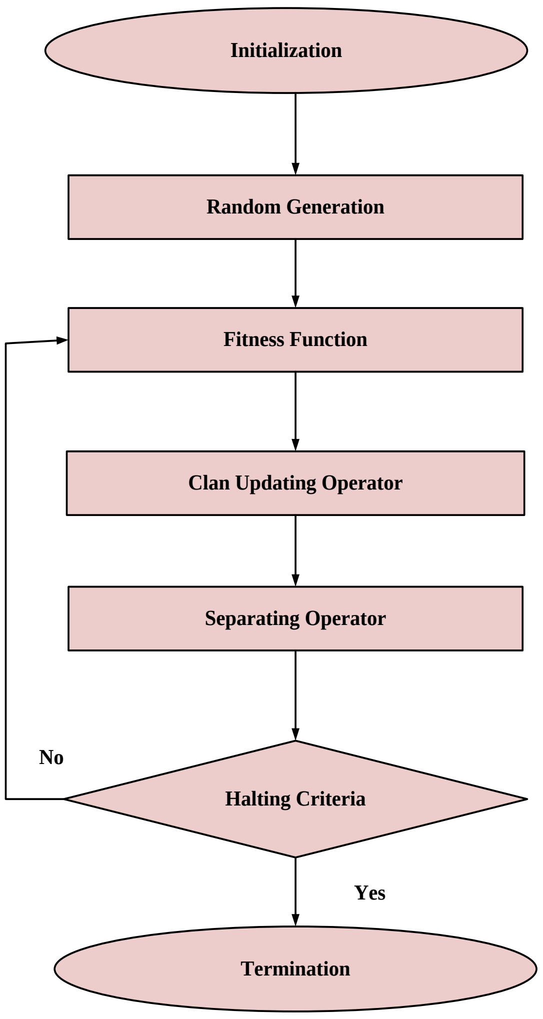

Fig.2. Flowchart for NIUS-EHOA

Step 4: Clan Updating Operator

It is recognized by their position in the search space. Assume that, an elephant clan specifies d_i. Next position contains several elephants j, the clan is updated by equation (11),

$$a_{new,di,j} = a_{di,j} + \alpha \times \left(a_{best,di} - a_{di,j}\right) \times s \tag{11}$$

where in the clan d_i , $a_{new,di,j}$ specifies the new position for individual j, $a_{best,di}$ denotes the best solution in clan d_i that is founded at this time, in clan d_i , $a_{di,j}$ specifies the old position of the individual j. Parameters such as $\alpha \in [0,1]$ specifies the scale indicator which designates the authority for matriarch ci on $a_{di,j}$, $s \in [0,1]$ specifies random variable with uniform distribution. Every clan d_i , updates the fittest solution and is shown in equation (12)

$$a_{new,di,j} = \beta \times a_{center,di} \tag{12}$$

where $\beta \in [0,1]$ represents the factor that impact $a_{center,di}$ on the updated individual, C denotes the average dimension of search space which follows the calculation of center clan d_i, $a_{center,di,c}$ for c^{th}dimension problem is shown in equation (13)

$$a_{center,di,c} = \frac{1}{n_{d,}i} \times \sum_{j=1}^{c} a_{di,j,c} \tag{13}$$

where $1 < a_{center,di} < c, n_d i$ specifies number of elephants in clan d_i, a_{di}, j, c specifies the c^{th} elephant individual a_{di}, j.

Step 5: Separating Operator

Here, the worst member of every clan is to move away the clan to discover better new positions of the tasks and is shown in equation (14)

$$a_{wrost,di} = aminmax_{min} \tag{14}$$

where a_{max} and a_{min} specifies the upper, lower bound for the positions of individual. $a_{wrost,di}$ specifies the worst individuals of clan d_i, and $rand \in [0,1]$ denotes the numbers which are chosen randomly by uniform distribution.

Step 6: Termination

The allocation of Virtual Machines (VMs) to the incoming tasks with the help of NIUS-EHOA will repeat step 3 iteratively until fulfil the halting criteria.

4. Results and Discussion

Here, experimental result is discussed for an NIUS-EHOA for effective balancing of load in cloud environments. Simulation process is done by Clouds API 3.0.3. Here the assessment matrices, MRT under various numbers of tasks, MRT under various executable instruction lengths, Mean Execution Time under different count of tasks and migrated tasks with different numbers of virtual machine are analyzed. Then the proposed NIUS-EHOA-LB-CE method is examined with existing methods like IABC-MBOA-LB-CE [20], an improved Hybrid FACOA-LB-CE [21], Hybridization of FF-IMOPSO-LB-CE [22] and A hybrid GWO-PSO-LB-CE [23]. Table 1 shows the simulation parameters setup that is considered for the implementation of NIUS-EHOA-LB-CE system

Table 1. Simulation parameters

Category	Parameter	Cost
Cloudlets	Count of cloudlets	100-1000
	Task Distance	2000-20000
Data center	VM scheduler	Time-Shared
	No. of hosts	2-10
	No. of data centers	20
Virtual Machine (VM)	Cloudlet Scheduler	Time-Shared
	Bandwidth	500-1200
	Necessary No. of processor elements	1-4
	Processor speed	4000-8000 MIPS
	No. of virtual machines	50
	Memory space available in each virtual machine	256-2018 Mb

Here, the potential of NIUS-EHOA-LB-CE system is discovered related to MRT under different number of tasks, MRT under various executable instruction lengths, Mean Execution Time below different number of tasks, migrated tasks among various numbers of virtual machine is compared with existing methods like IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively.

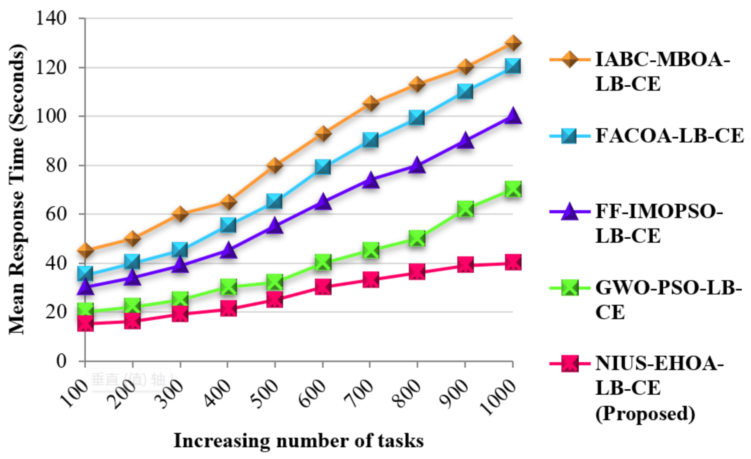

Fig.3. MRT under different count of tasks

The performance metrics of MRT under different count of tasks is depicted in Fig. 3. Here, NIUS-EHOA-LB-CE method is compared with existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-

CE methods. At task 200, the MRT of NIUS-EHOA-LB-CE method attains 18.75%, 24.05%, 56.22% and 31.25% lesser than the existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At task 400, the MRT for NIUS-EHOA-LB-CE method contains 36.77%, 25.85%, 35.22% and 11.05% lower than the existing method such as IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At task 600, the MRT for NIUS-EHOA-LB-CE method contains 9.66%, 29.05%, 16.22% and 11.05% lower than the existing method such as IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At task 800, the MRT for NIUS-EHOA-LB-CE method contains 22.75%, 34.05%, 16.22% and 51.16% lower than the existing method such as IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At task 1000, the MRT for NIUS-EHOA-LB-CE method contains 28.16%, 13.11%, 29.33% and 33.5% lower than the existing method such as IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively.

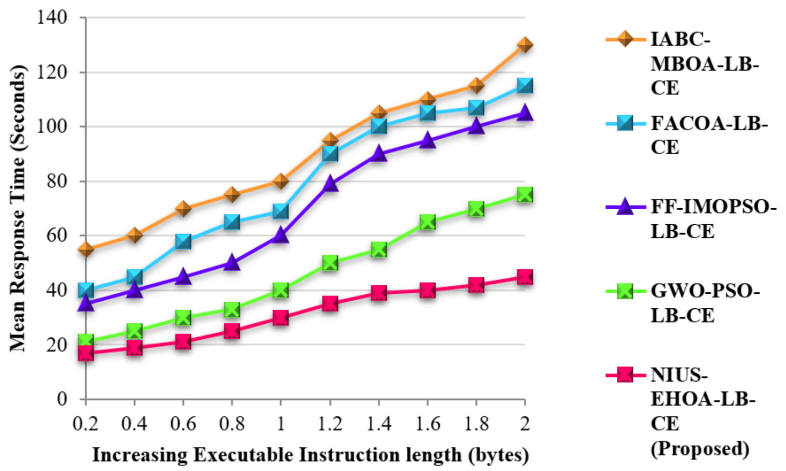

Fig.4. Mean response time under different executable instruction lengths

Fig. 4 shows MRT under various executable instruction lengths. Here the proposed NIUS-EHOA-LB-CE method is compared with existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE methods. At instruction length 0.4, the MRT for NIUS-EHOA-LB-CE method contains 46.77%, 19.16%, 26.32% and 11.55% lesser than the existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At instruction length 0.8, the MRT for NIUS-EHOA-LB-CE method contains 76.47%, 59.02%, 36.22% and 40.33% lesser than the existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At instruction length 1.2, the MRT for NIUS-EHOA-LB-CE method contains 6.07%, 11.16%, 36.32% and 45.55% lesser than the existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At instruction length 1.6, the MRT for NIUS-EHOA-LB-CE method contains 16.44%, 11.11%, 9.35% and 29.41% lower than the existing method such as IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At instruction length 2, the MRT for the proposed NIUS-EHOA-LB-CE method contains 77.46%, 16.6%, 22.33% and 31.45% lesser than the existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively.

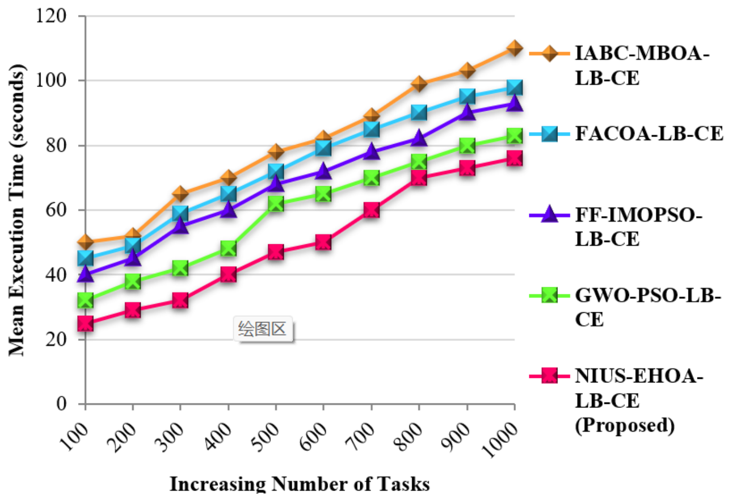

Fig.5. Mean execution time under different number of tasks

Fig. 5 depicts the efficacy metrics of MET under different number of tasks. Here, NIUS-EHOA-LB-CE method is compared with existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE methods. At task 200, the MET for NIUS-EHOA-LB-CE method contains 16.86%, 15.36%, 22.32% and 61.01% lower than the existing method such as IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At task 400, the MET for NIUS-EHOA-LB-CE method contains 15.77%, 21.33%, 12.33% and 9.56% lower than the existing method such as IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At task 600, the MET for the proposed NIUS-EHOA-LB-CE method contains 41.59%, 16.69%, 23.33% and 21.74% lower than the existing method such as IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At task 800, the MET for NIUS-EHOA-LB-CE method contains 27.26%, 11.26%, 29.33%, 32.91% lesser than the existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE models respectively. At task 1000, the MET for NIUS-EHOA-LB-CE method contains 66.74%, 10.38%, 29.57% and 51.21% lower than the existing method such as IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively.

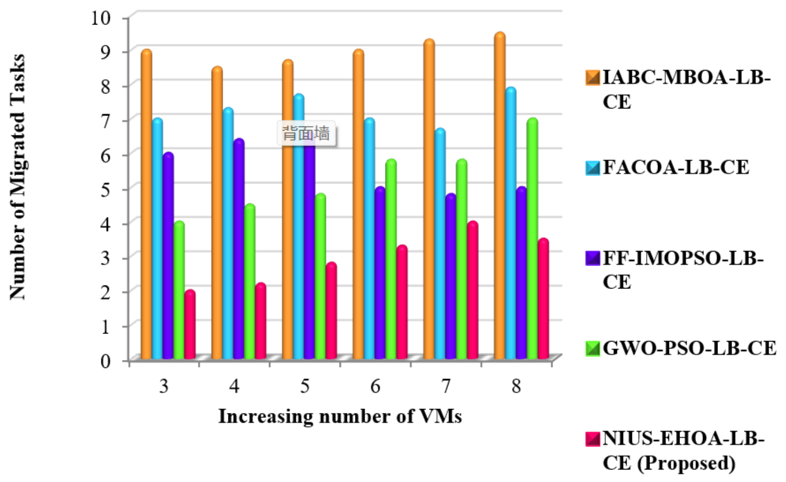

Fig.6. Count of migrated tasks with various count of VMs (count of tasks=500)

The performance metrics for the numbers of migrated task with different numbers of virtual machine (number of tasks=500) is shown in Fig. 6. Here the proposed NIUS-EHOA-LB-CE method is compared with existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE methods. With the increasing count of virtual machine 4, number of migrated tasks for NIUS-EHOA-LB-CE method contains 6.56%, 5.33%, 12.59% and 39.33% lesser than the existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE models respectively. At increasing number of virtual machines 6, the number of migrated tasks for NIUS-EHOA-LB-CE method contains 14.12%, 43.33%, 32.55% and 49.23% lower than the existing method such as IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At increasing number of virtual machines 8, the number of migrated tasks for NIUS-EHOA-LB-CE method is 9.12%, 13.33%, 25.06% and 49.32% lower than the existing method such as IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively.

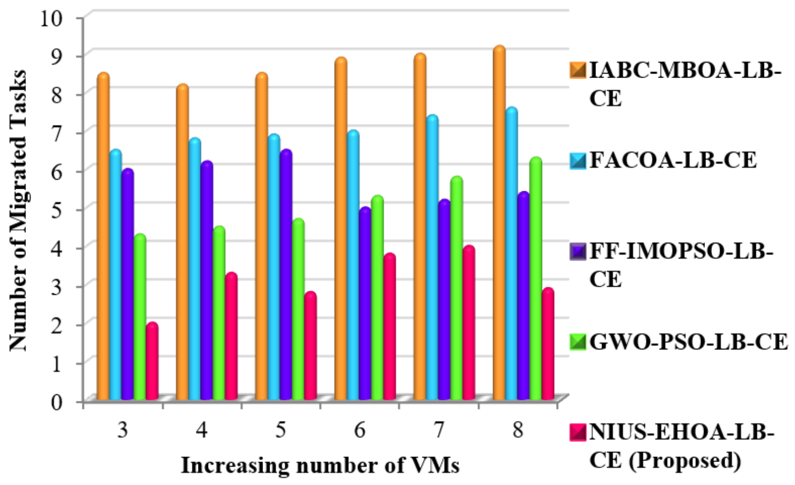

Fig.7. Number of migrated tasks with different number of VMs (number of tasks=1000)

Fig. 7 depicts the efficacy metrics for the numbers of migrated task with different numbers of virtual machine (task numbers =1000). Here the proposed NIUS-EHOA-LB-CE method is compared with existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE methods. With the increasing count of virtual machine 4, number of migrated tasks for NIUS-EHOA-LB-CE method is 4.16%, 9.87%, 16.53% and 29.42% lower than the existing method such as IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At increasing number of virtual machines 6, the number of migrated tasks for NIUS-EHOA-LB-CE method is 6.56%, 5.33%, 12.59% and 39.33% lesser than the existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE models respectively. At increasing number of virtual machines 8, number of migrated tasks for NIUS-EHOA-LB-CE method contains 9.55%, 15.23%, 11.59% and 33.33% lesser than the existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE models respectively.

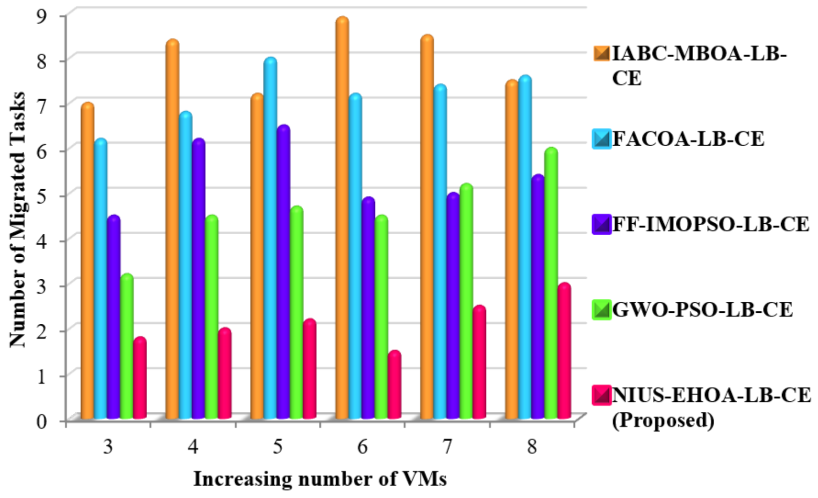

Fig.8. Mean response time under different number of tasks at 500

Fig 8 shows the performance metrics of MRT under various count of tasks at 500. Here the NIUS-EHOA-LB-CE method is compared with existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE method. With the increasing count of virtual machine 4, numbers of migrated task for NIUS-EHOA-LB-CE method of about 16.44%, 8.97%, 53.12% and 24.95% lower than the existing method such as IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At increasing number of virtual machines 6, numbers of migrated task for NIUS-EHOA-LB-CE method of about 8.12%, 7.87%, 13.56% and 34.15% lower than the existing method such as IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At increasing number of virtual machines 8, numbers of migrated task for NIUS-EHOA-LB-CE method of about 46.14%, 7.87%, 13.52% and 55.05% lesser than the existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE models respectively.

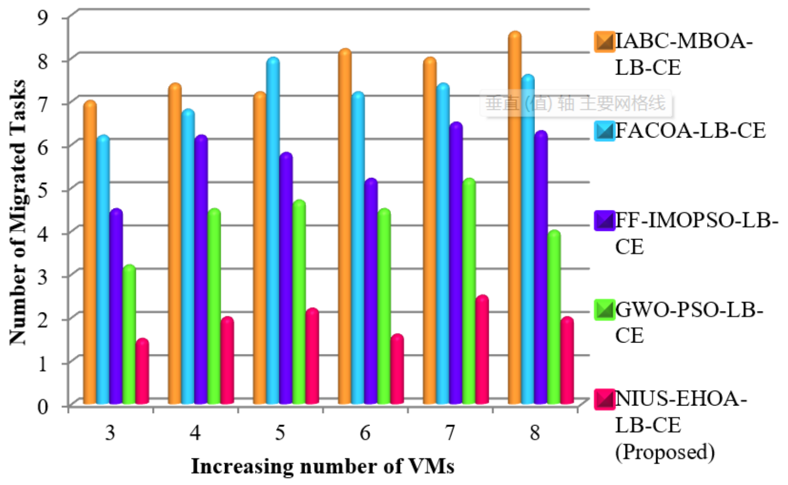

Fig.9. Mean response time under different number of tasks at 1000

Fig. 9 depicts the performance metrics of MRT under various count of tasks at 1000. Here the NIUS-EHOA-LB-CE method is compared with existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE methods. At increasing number of virtual machines 4, number of migrated tasks of proposed NIUS-EHOA-LB-

CE method is 44.66%, 16.87%, 33.52% and 14.45% lesser than existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE models respectively. At increasing number of virtual machines 6, the numbers of migrated task of NIUS-EHOA-LB-CE method consists of 13.12%, 19.15%, 23.02% and 15.44% lower than the existing method such as IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE respectively. At increasing number of virtual machines 8, number of migrated tasks of proposed NIUS-EHOA-LB-CE method is 54.06%, 11.19%, 22.02% and 18.33% lesser than existing IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE models respectively.

Finally, Tables 2-4 represents the importance of proposed NIUS-EHOA-LB-CE system evaluates using make span, imbalance degree, numbers of task are migrated as a visualized before, after process of load balancing.

Table 2. Makespan (seconds) identified prior and after LB

Tasks	**IABC-MBOA-LB-CE**		**FACOA-LB-CE**		**FF-IMOPSO-LB-CE**		**GWO-PSO-LB-CE**		**NIUS-EHOA-LB-CE (Proposed)**	
200	25.13	13.63	23.34	11.65	21.38	10.15	18.23	9.45	11.6	5.54
400	27.67	16.45	26.78	13.36	25.68	12.12	23.45	11.35	18.54	7.32
600	32.67	18.52	31.24	14.68	29.78	13.46	25.46	12.12	21.45	8.16
800	33.39	20.31	32.56	14.98	30.24	13.48	28.56	12.82	28.78	11.76
1000	35.12	21.34	34.68	15.68	32.86	14.78	30.62	13.12	37.23	12.36

Table 3. Imbalance degree is identified before and after LB

Tasks	**IABC-MBOA-LB-CE**		**FACOA-LB-CE**		**FF-IMOPSO-LB-CE**		**GWO-PSO-LB-CE**		**NIUS-EHOA-LB-CE (Proposed)**	
200	1.98	0.56	1.88	0.44	1.78	0.38	1.63	0.34	1.56	0.21
400	2.34	0.58	1.96	0.52	1.83	0.45	1.67	0.37	1.59	0.23
600	2.76	0.62	2.12	0.59	1.89	0.48	1.69	0.39	1.64	0.25
800	2.89	0.65	2.18	0.65	1.97	0.54	1.72	0.43	1.68	0.27
1000	2,92	0.68	2.23	0.72	2.12	0.58	1.78	0.47	1.69	0.30

Table 4. Number of tasks identified to be migrated before and after LB

Tasks	**IABC-MBOA-LB-CE**		**FACOA-LB-CE**		**FF-IMOPSO-LB-CE**		**GWO-PSO-LB-CE**		**NIUS-EHOA-LB-CE (Proposed)**	
200	8.95	5.55	8.88	4.24	8.21	3.98	7.34	3.21	5.32	2.12
400	9.35	5.67	9.02	4.56	8.28	4.04	7.38	3.26	5.38	2.16
600	9.59	6.23	9.12	4.58	8.38	4.08	7.45	3.39	5.46	2.24
800	9.77	6.45	9.19	4.64	8.46	4.14	7.54	3.45	5.57	2.28
1000	9.81	6.89	9.28	4.69	8.56	4.19	7.59	3.48	5.63	2.34

The results confirmed that, the make span is attained before, after load balancing through NIUS-EHOA-LB-CE system developed at an average of about 5.24%, 6.24%, 7.34% and 7.94% compares with the existing approaches like IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE. Degree of imbalance is attained before, after the load balancing through NIUS-EHOA-LB-CE system improved at an average of about 4.82%, 5.82%, 10.23%, 7.84% compare the benchmarked IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE methods. Additionally, numbers of task migrated before, after load balancing through NIUS-EHOA-LB-CE system also minimized as an average of 6.74%, 7.16%, 11.25% and 9.28% compared to the benchmarked IABC-MBOA-LB-CE, FACOA-LB-CE, FF-IMOPSO-LB-CE and GWO-PSO-LB-CE approaches.

5. Conclusions

Here in this manuscript, an Individual Updating Strategies based EHOA is successfully implemented to facilitate the effective load balancing process. In this proposed NIUS-EHOA-LB-CE concentrated on the process of distributing the workloads associated with each network links for the attaining the objective of mitigating over-utilization and under-utilization of the cloud resources. The efficiency of NIUS-EHOA-LB-CE is assessed in terms of MRT under various numbers of tasks; MRT under various executable instruction lengths, MET under various numbers of tasks, numbers of migrated task with different numbers of virtual machines is analyzed. Experimental outcomes show that, the NIUS-EHOA-LB-CE method shows lower MRT under different number of tasks at 500 11.23%, 14.28%, 20.31% and 19.13%, lower MRT under different number of tasks at 1000 10.56%, 13.42%, 12.39% and 18% compared with existing IABC-MBOA-LB-CE, An improved Hybrid FACOA-LB-CE, Hybridization of FF-IMOPSO-LB-CE and A hybrid GWO-PSO-LB-CE methods. The task migration from the containers interfered with communication between the containers belonging to the same host, according to the experimental analysis. With the use of deep learning ideal path approaches, this issue will continue to be solved in the future.

References

[1] S.K. Mishra, B. Sahoo, and P.P. Parida, Load balancing in cloud computing: a big picture. *Journal of King Saud University-Computer and Information Sciences*, vol.32, no.2, pp.149-158.2020.

[2] K. Ramya, and S. Ayothi, "Hybrid dingo and whale optimization algorithm-based optimal load balancing for cloud computing environment." *Transactions on Emerging Telecommunications Technologies*, vol.34, no.5, p.e4760. 2023.

[3] S. Jeddi, and S. Sharifian,"A hybrid wavelet decomposer and GMDH-ELM ensemble model for Network function virtualization workload forecasting in cloud computing." *Applied Soft Computing*, vol.88, p.105940.2020

[4] W. Wang, H.Guo, X. Li, S. Tang, Y. Li, L.Xie, and Z. Lv, "Journal of Industrial Information Integration." *Journal of Industrial Information Integration*, vol.21, p.100189.2021

[5] Z.Royaee, H. Mirvaziri, and A. KhatibiBardsiri, "Designing a context-aware model for RPL load balancing of low power and lossy networks in the internet of things." *Journal of Ambient Intelligence and Humanized Computing*, vol.12, pp.2449-2468.2021

[6] R. Bhaskar, and B.S. Shylaja, "Dynamic Virtual Machine Provisioning in Cloud Computing Using Knowledge-Based Reduction Method." In *Next Generation Information Processing System: Proceedings of ICCET 2020, Volume 2* (pp. 193-202). Springer Singapore.2021

[7] S.M. Mirmohseni, A Javadpour, and C. Tang, "LBPSGORA: create load balancing with particle swarm genetic optimization algorithm to improve resource allocation and energy consumption in clouds networks." *Mathematical Problems in Engineering*, 2021, pp.1-15.2021

[8] F. Ebadifard, and S.M. Babamir, "Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment." *Cluster Computing*, *24*, pp.1075-1101.2021

[9] Z. Miao, P. Yong, Y. Mei, Y.Quanjun, and X. Xu, "A discrete PSO-based static load balancing algorithm for distributed simulations in a cloud environment." *Future Generation Computer Systems*, *115*, pp.497-516.2021

[10] Z .Tong, X. Deng, H. Chen, and J. Mei, "DDMTS: A novel dynamic load balancing scheduling scheme under SLA constraints in cloud computing." *Journal of Parallel and Distributed Computing*, *149*, pp.138-148.2021

[11] S.M. Ali, N. Kumaran, and G.N. Balaji, "A hybrid elephant herding optimization and harmony search algorithm for potential load balancing in cloud environments." *International Journal of Modeling, Simulation, and Scientific Computing*, vol.13, no.05, 2250042. 2022.

[12] V. Meyer, D.F. Kirchoff, Da M.L. Silva, and De C.A Rose, "ML-driven classification scheme for dynamic interference-aware resource scheduling in cloud infrastructures." *Journal of Systems Architecture*, *116*, p.102064.2021

[13] J. Tang, G. Liu, and Q. Pan, "A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends. *IEEE/CAA Journal of AutomaticaSinica*, vol.8(10), pp.1627-1643.2021

[14] O.Y. Abdulhammed, "Load balancing of IoT tasks in the cloud computing by using sparrow search algorithm." *The Journal of Supercomputing*, *78*(3), pp.3266-3287.2022

[15] Sim Sze Yin, Yoni Danieli. "A Hybrid Optimization Algorithm on Cluster Head Selection to Extend Network Lifetime in WSN",*Journal of Computing in Engineering*, 2020

[16] M. Sohani, and S.C. Jain, "A predictive priority-based dynamic resource provisioning scheme with load balancing in heterogeneous cloud computing." *IEEE access*, *9*, pp.62653-62664.2021

[17] D.A Shafiq, N.Z. Jhanjhi, A. Abdullah, and M.A.Alzain, "A load balancing algorithm for the data centres to optimize cloud computing applications." *IEEE Access*, vol.9, pp.41731-41744.2021

[18] J.P.B. Mapetu, L Kong, and Z Chen, "A dynamic VM consolidation approach based on load balancing using Pearson correlation in cloud computing." *The Journal of Supercomputing*, *77*, pp.5840-5881.2021

[19] J. Li, L.Guo, Y Li, and C Liu, "Enhancing elephant herding optimization with novel individual updating strategies for large-scale optimization problems." *Mathematics*, vol.7, no.5, p.395.2019

[20] S. Janakiraman, and M.D. Priya, "Improved artificial bee colony using monarchy butterfly optimization algorithm for load balancing (IABC-MBOA-LB) in cloud environments." *Journal of Network and Systems Management*, *29*(4), p.39.2021

[21] A. Ragmani, A. Elomri, N. Abghour, K. Moussaid, and M. Rida, "An improved hybrid fuzzy-ant colony algorithm applied to load balancing in cloud computing environment." *Procedia Computer Science*, *151*, 519-526. 2019.

[22] A.F.S.Devaraj, M. Elhoseny, S.Dhanasekaran, E.L. Lydia, and K Shankar, "Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments." *Journal of Parallel and Distributed Computing*, *142*, pp.36-45.2020.

[23] B.N. Gohil, and D.R. Patel, "A hybrid GWO-PSO algorithm for load balancing in cloud computing environment." In *2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT)* (pp. 185-191). IEEE.2018

[24] U.K. Jena, P.K. Das, and M.R. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment." *Journal of King Saud University-Computer and Information Sciences*, vol.34, no.6, pp.2332-2342.2022

[25] D.A. Shafiq, N.Z. Jhanjhi, and A. Abdullah, "Load balancing techniques in cloud computing environment: A review." *Journal of King Saud University-Computer and Information Sciences*, vol.34, no.7, pp.3910-3933.2022

[26] N Joshi, K.Kotecha, D.B. Choksi, and S. Pandya, "Implementation of novel load balancing technique in cloud computing environmen." In *2018 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1-5). IEEE.2018

[27] V. Polepally, and K. Shahu Chatrapati, "Dragonfly optimization and constraint measure-based load balancing in cloud computing". *Cluster Computing*, vol.22(Suppl 1), pp.1099-1111. 2019.

Authors' Profiles

Syed Muqthadar Ali received B.Tech degree in CSE from AnwarulUloom College of Engineering & Technology, Vikarabad, affiliated to Jawaharlal Nehru Technological University, Hyderabad, A.P. in 2003. M.Tech Degree in CSE from CVR College of Engineering, Ibrahimpatnam, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad, Telangana in 2013. He is currently working toward the Ph.D. degree at the Department of Computer Science & Engineering, Annamalai University, Chidambaram, Tamil Nadu, India. His research interests include Cloud Computing.

N. Kumaran received B.E degree in CSE from Adhiparasakthi Engineering College, Melmaruvathur, affiliated to University of Madras, Chennai, Tamil Nadu in 2002. M.E Degree in CSE from Annamalai University, Chidambaram, Tamil Nadu in 2004. Ph.D. Degree in CSE Annamalai University, Chidambaram, Tamil Nadu in 2016. He is currently working as an Assistant Professor at the Department of Computer Science Engineering, Annamalai University, Chidambaram, Tamil Nadu, India. His research interests include Image Processing, Medical Imaging, Genetic Algorithm, Soft Computing, CBIR and Cloud Computing. He has published more than 12 international papers & 12 papers in national and international conferences.

G.N. Balaji received B.E degree in CSE from Annamalai University, Chidambaram in 2010. M.Tech Degree in CSE from SRM University, Chennai, Tamil Nadu in 2012. Ph.D. Degree in CSE Annamalai University, Chidambaram, Tamil Nadu in 2017. He is currently working as an Associate Professor, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India. His research interests include Image Processing and Machine Learning. He has published more than 30 research papers in peer reviewed international journals and conferences.

How to cite this paper: Syed Muqthadar Ali, N. Kumaran, G.N. Balaji, "Individual Updating Strategies-based Elephant Herding Optimization Algorithm for Effective Load Balancing in Cloud Environments", International Journal of Computer Network and Information Security(IJCNIS), Vol.16, No.2, pp.65-78, 2024. DOI:10.5815/ijcnis.2024.02.06