International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Hybrid Elephant Herding Optimization and Flamingo Search Algorithm for Effective load Balancing in Cloud Computing

Mr. Syed Muqthadar Ali[1*], Dr. N. Kumaran[2], Dr. G. N. Balaji[3]

Submitted: 27/04/2023 **Revised**: 28/06/2023 **Accepted**: 06/07/2023

Abstract: Cloud computing has many challenges, such as server failures, loss of confidentiality, improper workloads still limit the performance of cloud systems in real-world scenarios. Due to this, numerous research works are being carried out to improve the limitation of existing systems. Among them, load balancing seems to be a major issue that degrades the performance of the cloud industry, so optimal load balancing with optimal task scheduling is required. With the aim of attaining optimal load balancing by efficacious task deployment, in this manuscript Hybrid Elephant Herding Optimization and Flamingo Search Algorithm is proposed for effectual load balancing in cloud environment (LBS-CE-Hyb-EHO-FSA). The aim of proposed LBS-CE-Hyb-EHO-FSA is to enhance the population initialization and search space exploitation for activating the predominant load balance among the virtual machines (VMs) in the clouds. It includes the weighted task scheduling procedure depending on the optimization issue formulated utilizing the parameters of makespan, energy consumption and data center cost. Here, LBS-CE-Hyb-EHO-FSA is proposed for exploiting the merits of Elephant Herding Optimization (EHO) algorithm and Flamingo Search Algorithm (FSA) in order to achieve superior results in all dimensions of cloud computing. In this, LBS-CE-Hyb-EHO-FSA achieves the allocation of Virtual Machines (VMs) to incoming tasks of cloud, when the number of currently processing tasks of a specific VM is reduced than cumulative number of tasks presently processing by other VMs in the cloud. It also attains potential load balancing process, then difference between the processing time of all individual virtual machine and the mean response time (MRT) incurred by the complete virtual machine. Finally, the simulation experiment of proposed LBS-CE-Hyb-EHO-FSA is conducted using Cloudsim platform. Here the proposed method provides 23.35%, 15.06%, 21.77%, 27.82%, 14.31%, 19.23% lower Mean Execution Time and 38.22%, 40.21%, 19.30%, 25.46%, 19.25%, 21.14% lower mean response time comparing to the existing models.

Keywords**:** *Cloud environment, Elephant herd optimization, Flamingo Search Algorithm, Load balancing, Mean response time*.

1. Introduction

Generally, the cloud computing is a significant role for facilitating as a game-changer in most of the operations that involves resource intensive applications, such as operating modes, collaborative capacities, end-user services, service provisioning [1]. The main aim of the cloud services is to provide end-users with quick access to the virtual machines. The load balancing is focused on upgrading the performance with minimal cost and energy consumption [2]. When raised the count of overloaded VMs in the network, the performance of the cloud environment is substantially lessened [3-5]. Outages and unavailability of tasks are also caused by overloaded VMs, resulting in a decline in the degree of system usage in public cloud. Nearly, 60% energy consumes through the data centres that keep idle server.[6, 7] The advantage of cloud computing is a well-organized load balancing tactic contains low waiting time, propinquity, real time interaction and occupy more. Also, its disadvantage is energy consumption, load balancing rate and delay. These drawbacks motivated to do this research work. In this manuscript, the Hybrid Elephant Herding Optimization and Flamingo Search Algorithm are proposed for efficient Load Balancing in the cloud environment.[8-10] This, LBS-CE-Hyb-EHO-FSA is proposed to exploit the merits of traditional EHO algorithm and FSA, in order to achieve superior results in all dimensions of cloud computing. This proposed LBS-CE-Hyb-EHO-FSA prevented the shortcomings of the existing metaheuristic algorithms in attaining superior load balance between the physical machines.

The key contributions of this manuscript are described below,

- Hybrid Elephant Herding Optimization (HEHO) [11] and Flamingo Search Algorithm (FSA) [12] are both metaheuristic optimization algorithms that have been used in various applications such as job scheduling, task allocation, and load balancing. When combined, HEHO and FSA can provide an efficient solution for load balancing in a cloud environment.

[1*]*Research Scholar, Department of Computer Science and Engineering, Annamalai University. Annamalainagar, Chidambaram 608002, Tamil Nadu, India*
[1*]*Email: smuqthadarali34@gmail.com*
[2]*Assistant Professor, Department of Computer Science and Engineering, Annamalai University. Annamalainagar, Chidambaram 608002, Tamil Nadu, India*
[2]*Email: kumaran81@gmail.com*
[3]*Associate Professor, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India*
[3]*Email:balaji.gnb@gmail.com*
** Corresponding Author Email: smuqthadarali34@gmail.com*

- Cloud computing is a popular paradigm for delivering computing resources as a service via the internet. One of the key challenges in a cloud environment is to ensure that the workload is evenly distributed across the available resources. Load balancing can help to optimize resource utilization, improve system performance, and ensure that the user's requirements are met.
- HEHO is inspired by the herding behavior of elephants and uses a combination of local and global search strategies to find optimal solutions. FSA, on the other hand, is based on the flocking behavior of flamingos and uses a collective intelligence approach to find the best solution. By combining the strengths of both algorithms, a hybrid approach can be developed that is more robust and effective than either algorithm alone.
- The HEHO-FSA hybrid algorithm can be used for load balancing in a cloud environment by considering the workload distribution across the available resources. The algorithm can be used to optimize the allocation of tasks or virtual machines to different nodes, based on factors such as the CPU utilization, memory usage, and network traffic. By using HEHO and FSA together, the algorithm can quickly converge to an optimal solution while avoiding local optima.
- In summary, the HEHO-FSA hybrid algorithm can provide an efficient solution for load balancing in a cloud environment by combining the strengths of both algorithms. The algorithm can help to optimize resource utilization, improve system performance, and ensure that the user's requirements are met.

Rest of the manuscript is described as; section 2 specifies the literature review of different researches related to efficient Load Balancing of Cloud Environment. Section 3 defines the proposed methodology. Section 4 represents the results and discussion. At last, section 5 concludes the manuscript.

2. Literature Review

Various research works were previously presented in the literature associated to effective load balancing process in cloud computing. A few research works are reviewed in this section, Kaur, and Kaur, [13] have suggested a hybrid heuristic-metaheuristic based load balancing optimization in cloud environment. It attains higher mean response time. Balaji et al., [14] have presented LBS-CE-ACSO. Here, the presented algorithm increases the system resource usages of virtual machine and decrease the usage of power. For load balancing, it exhibits reduced energy utilization, conversely, the memory utilization is higher. Devaraj et al., [15] have presented the hybridization of firefly and enhanced multiple objective particle swarm optimization for energy efficient load balancing in cloud computing environments. It provides least average response time with higher makespan. Prassanna and Venkataraman, [16] have presented an adaptive regressive holt–winters workload prediction with firefly optimized lottery scheduling to load balance in cloud environment. It provides better task scheduling performance with higher makespan. Ziyath, and Senthilkumar, [17] have presented meta-heuristic optimization applied task scheduling with load balancing technique for cloud infrastructure services. It provides minimum energy consumption and higher data center cost.

3. Proposed Methodology

Here, the Hybrid elephant herding optimization algorithm and the Flamingo Search Algorithm are proposed for facilitates the effective load balancing process in cloud computing (LBS-CE-Hyb-EHO-FSA). The block diagram of proposed LBS-CE-Hyb-EHO-FSA method is given in Figure 1. The detailed discussion regarding LBS-CE-Hyb-EHO-FSA are given below,

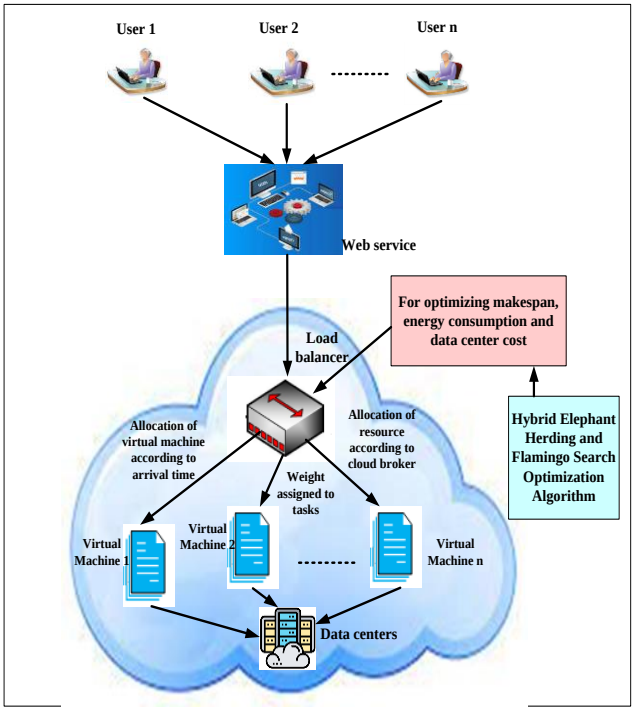

Fig 1: Block diagram of proposed LBS-CE-Hyb-EHO-FSA method

3.1 System model

The system model employed to implement the proposed LBS-CE-Hyb-EHO-FSA scheme contains 'k' hosts or cloud data centres represents set $C =: \{c_1, c_2, ...c_k\}$ including 'm' count of VMs is signified using $V_M = \{V_{M(1)}, V_{M(2)}, ...V_{M(r)}\}$ and 'm' represents the count of tasks that is portrayed through $T_S = \{T_{S(1)}, T_{S(2)}, ..., T_{S(m)}\}$. The tasks submit to cloud computing platform using the accessible cloud brokers.

These tasks submit to the clouds and it is emphasized by a collection of factors $t_{s(i)} = \{tft_i, tte_i.tl_i, ta_i\}$ which presents task final stage, threshold time essential for task implementation, length and arrival time on cloud environment. The entire collection of factors is transmitted into $V_{M(i)}$. The proposed LBS-CE-Hyb-EHO-FSA method focus mainly on under and over-utilization virtual machines, tasks' makespan, datacentre cost and energy consumes, '$PT_{Task(i)}$' emphasis the implementation time of tasks utilizing Equation (1).

$$PT_{Task(m)} = \sum_{i=1}^{m} S_{ij} \tag{1}$$

where $1 \leq j \leq n$

where S_{ij} denotes the number of processors. The capacity of virtual machine $VM_{CAP(j)}$ depending on bandwidth ($BW_{PE(j)}$), million instructions/sec($MIPS_{PE(j)}$), processing count ($PC_{PE(j)}$) corresponding to the processing components of clouds is computed using Equation (2).

$$VM_{CAP(j)} = BW_{PE(j)} * MIPS_{PE(j)} * PC_{PE(j)} \tag{2}$$

The load has feasibility allocated to every virtual machine $LOAD_{VM}$ is scaled using Equation (3).

$$LOAD_{VM} = \frac{Tasks_{TN}(T,t)}{SR(V_{M(i)},t)} \tag{3}$$

In Equation (3), total number of tasks $TN_{Tasks}(T,t)$ and service rate $SR(V_{M(i)},t)$ of VMs at time t . The sum of task load allocated to the entire set of VMs activating at the cloud utilizing Equation (4).

$$LOAD_{C(VM)} = \sum_{j=1}^{n} LOAD_{VM(j)} \tag{4}$$

where $LOAD_{VM(j)}$ represents the count of loads in VM, n denotes the count of tasks. The time incurred for processing the tasks is submitted to the total count of virtual machines exist in the cloud environment, and it is presented in Equation (5)

$$TP_{TOTAL\_VM)} = \frac{LOAD_{C(VM(j))}}{CAP_{VM(j)}} \tag{5}$$

where $CAP_{VM(j)}$ denotes the virtual machine capacity, $TP_{TOTAL\_VM)}$ represents the processing time of VM. The execution of every task is allocated to individual virtual machines computed utilizing Equation (6), (7).

$$Time_{Exec} = \frac{Tl_{(i)}}{CPU_{Fract(i)}} \tag{6}$$

$$TP_{(I-VM)} = \frac{LOAD_{VM(i)}}{CAP_{VM(j)}} \tag{7}$$

where $Time_{Exec}$ specifies the task execution time, $Tl_{(i)}$ denotes the average length of tasks that is submitted to the particular virtual machine, $CPU_{Fract(i)}$ represents the fractional capacity, $TP_{(I-VM)}$ denotes the processing time of VM. The ending time of individual task allocates to particular virtual machine is computed in terms of execution time as well as starting time receiving to the clouds depending on Equation (8).

$$TF_{TS(n)} = STime_{TS(n)} + Time_{Exec} \tag{8}$$

where $TF_{TS(n)}$ represents the finishing time of tasks, $STime_{TS(n)}$ represents the execution start time. The decision variable $DS_{VAR(ij)}$ deemed to allocate the incoming tasks to the associated virtual machines utilizing tasks processing time labelled in Equation (9).

$$DS_{VAR(ij)} = \begin{cases} 1 & if \quad tft_i < tte_i \\ 0 & if \quad tft_i > tte_i \end{cases} \tag{9}$$

here the makespan means the overall time to complete the task with respect to proficient allocation of virtual machines. This factor of makespan required to be optimally lessened. The objective fitness function focuses on lessening the tasks makespan enters inside the cloud computing environment and it is specified in Equation (10).

$$M_S = Min(\underset{s \in T_S, j \in VM_j}{Max} ft_{ij}) \tag{10}$$

Let ft_{ij} represents the time of finishing that is acquired by $T_{S(s)}$ task above the virtual machine $VM_{(j)}$, M_S represents the minimal makespan of task and $\underset{s \in T_S, j \in VM_j}{Max}$ denotes the makespan of the tasks incoming to cloud computing. If energy consumptions enters based on the task execution $T_{S(s)}$ above the virtual machine $VM_{(j)}$ represents $EC_{Task(s)}$ .The energy consumption rate ($Rate\_EC_{Task(s)}$) is acquired by the virtual machine with the time of task execution ($Exec_{Time}$) to the corresponding virtual machine is assessed by equation (11)

$$= EC_{Task(s)} = Rate\_EC_{Task(s)} * Exec_{Time} \tag{11}$$

The cumulative energy consumptions of each virtual machine are used for processing the tasks is assessed by Equation (12).

$$TEC(VM_{(j)}) = \sum_{i=1}^{k}\sum_{j=1}^{s} EC_{Task(s\_ij)} \quad (12)$$

where $TEC(VM_{(j)})$ denotes the cumulative energy consumption for each virtual machine. The objective function concentrates on the energy consumption utilizing Equation (13).

$$E_{Con} = Min(TEC(VM_{(j)}) \quad (13)$$

where E_{Con} represents the objective function for energy consumption and $Min(TEC(VM_{(j)})$ denotes minimization of cumulative energy consumptions. Additionally, the datacentre cost specifies other parameter is considered for task scheduling to the virtual machine is depend on its availability is determined by equation (14).

$$DC_{Cost}(VM_{(j)}) = VM_{(j)} * Cost_{PER-UNTI} \quad (14)$$

here $DC_{Cost}(VM_{(j)})$ denotes the data centre cost of VM and $Cost_{PER-UNIT}$ specifies the cost acquired to utilize one kilo watt energy using data center at the operations of cloud. Therefore, the objective function focuses less datacentre cost that is determined in equation (15).

$$D_{C\cos t} = Min(DC_{Cost}(VM_{(j)}) \quad (15)$$

At last, the above mentioned objective function is expressed as makespan (M_S), the energy consumption (E_{Con}), data center cost ($D_{C\cos t}$) are based on the below equation (16-18)

$$\sum_{i=1}^{s} \psi_{ij} = 1,$$

$$(t_{S(s)} \in T, VM_{(j)} \in V \quad (16)$$

In equation (16), the aforementioned constraints emphasizes a single task is needed to be assigned to every separate virtual machines.

$$\sum_{i=1}^{k} T_{S(i0} \le etd_i,$$

$$(t_{S(s)} \in T, VM_{(j)} \in V \quad (17)$$

In equation (17), the time for performing the task is lesser than complete deadline for the specific task through the virtual machine.

$$\sqrt{\frac{1}{k}\sum_{i=1}^{k}(PT_{Task(s\_i)} - PT_{Task(i)})^2} \le UP_{Th} \quad (18)$$

In equation (18), calculates the standard deviation of load is less than higher values of threshold in virtual machine allocation. Also, the load balancing procedure is depending on the degree of imbalance that is represented in Equation (19)

$$\mathrm{Im}b\_Degree = \frac{Max_{Task(s)} - Min_{Task(s)}}{Mean_{Task(s)}} \quad (19)$$

where $\mathrm{Im}b\_Degree$ denotes the imbalance degree, $Max_{Task(s)}$, $Min_{Task(s)}$ and $Mean_{Task(s)}$ specifies the maximal, minimal, mean count of tasks are presented in cloud environment.

3.2 Systematic steps for proposed LBS-CE-Hyb-EHO-FSA scheme

The LBS-CE-Hyb-EHO-FSA approach utilize the multi-objective function to decide the allocation and re-allocation of newer/older task with suitable virtual machine/host. After allocates the task to them, the allocation and reallocation depends upon primitive constraints for emphasizing the load of virtual machine is higher than the value of upper limit. If large number present in virtual machines, then determines the constraint of deadline. Additionally, the task migration from heavily and lightly loaded virtual machine is necessary for deadline/completion time. Here, select the virtual machine along minimal value of higher deadline task, when higher the completion time of receiving or re-allocating task. Besides, the virtual machines with higher as well as medium deadline tasks have chosen while the incoming or re-allocating task completion time is medium. The virtual machine group is totally depending on VM existing load. VMs present in over-loaded virtual machine group for removing the task and it waits till it identifies the potential virtual machine for the allocation on subsequent iterations. The virtual machines are allocated in under-loaded group for the waiting task that required to be reallocated.

3.3 Hybrid elephant herding optimization and Flamingo Search Algorithm based load balancing process

In cloud computing scenario, EHO and FSA is proposed to attain suitable balance of virtual machine along with the objective function depending on makespan (M_S), energy consume (E_{Con}), cost of data center ($D_{C\cos t}$) for LBS-CE-Hyb-EHO-FSA is delineated, its corresponding flow chart is specified in Figure 2. The hybridization of two metaheuristic algorithms such as Elephant Herding Optimization (EHO) and Flamingo Search Algorithm (FSA) can provide several advantages over using either algorithm alone. Some potential benefits of this hybridization include:

- **Improved Global Exploration:** EHO is known for its strong local search capabilities, while FSA is good at

global exploration. By combining the two, we can benefit from both algorithms' strengths to improve the search process, ensuring a more thorough exploration of the solution space.

- **Better Convergence:** Combining EHO and FSA can lead to faster convergence to optimal or near-optimal solutions. EHO's local search capability can help refine solutions found by FSA, leading to improved convergence.
- **Robustness:** The hybridization of EHO and FSA can increase the algorithm's robustness, making it less prone to getting trapped in local optima. The combination of different search strategies can help the algorithm escape local optima and find better solutions.
- **Applicability:** The hybridization of EHO and FSA can make the algorithm more versatile and applicable to a wider range of problems. EHO and FSA have been successfully applied to a variety of optimization problems in different fields. Combining them can help extend their applicability even further.

Overall, the hybridization of EHO and FSA can lead to an algorithm with better performance, faster convergence, improved robustness, and wider applicability.

Firstly, the hybrid EHO and FSA creates the uniform distribution initial population of elephant and flamingo. Then, the parameters are generated randomly, after the process of initialization, and it calculates the fitness function. While using the herd behaviour of elephants together with foraging behaviour of Flamingo optimizes the makespan (M_S), energy consume (E_{Con}), cost of data center ($D_{C\cos t}$) is used for efficient load balancing in the cloud environment. The optimum solution is updated by the Hybrid EHO and FSA. Then, the above mentioned procedure is repeated until met the feasible solution. The step-wise process is specified below,

Step 1: Initialization

The process of elephant and flamingo is initialized. The populations of the elephant and flamingos are considered as $q = 1,2,3,.....,M$ and the location of elephant and flamingos is considered as $n = 1,2,3,.....,N$, which are initialized.

Step 2: Random Generation

The input parameters randomly created after the initialization process. Hence, the values

of best fitness for each elephant and flamingo are selected based on the explicit hyper-parameter situation.

Step 3: Fitness Function

After initialized values, the arbitrary count of resolution is created. Then, fitness function is scaled using the given equation (20)

$$Fitness\ \ function = Minimize[Makespan(M_S), Energy\ \ Consume\ (E_{Con}), Data\ center(D_{C\cos t})] \quad (20)$$

Step 4: Herding behaviour of elephants for minimizing M_S and E_{Con}

In this step, for each elephant is recognized by their position in the search space. Assume that, an elephant clan specifies d_i . Next position consist of several elephant j in the clan is updated, using the below equation (21),

$$a_{new,di,j} = a_{di,j} + \alpha \times \left(a_{best,di} - a_{di,j}\right) \times s \quad (21)$$

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 , from the position updation, makespan (M_S) is decreased by using the equation (22)

$$M_S = \beta \times a_{center,di} \quad (22)$$

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 centre clan d_i , $a_{center,di,c}$ for c^{th} dimension problem for minimizing energy consumption E_{Con} is shown in equation (23)

$$E_{Con} = \frac{1}{n_d, i} \times \sum_{j=1}^{c} a_{di,j,c} \quad (23)$$

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*:** Foraging behaviour of flamingo for lessening $D_{C\cos t}$

Foraging behaviour of flamingo decreases is specified as $D_{C\cos t}$ and it is discussed in equation (24).

$$D_{C\cos t} = \left(X^i_{pq} + \varepsilon_1 . Xb^i_q + g_2 \times \left| g_1 . Xb^i_q + \varepsilon_2 . X^i_{pq} \right|\right) / K \quad (24)$$

where the position of the channel from (p^{th}) secondary transmitter to (q^{th}) secondary receiver epitomizes X_{pq}^{i+1} in the $(i+1)$ iterations, X_{pq}^{i} epitomizes position of channel in i^{th} iteration, K depicts diffusion factor. Also, $\varepsilon_1, \varepsilon_2$ indicates random number [-1 or 1], g_1, g_2 are the random value as [0,1] is followed by the standard normal distribution factor. Additionally, the best fitness value is mentioned as Xb_q^i of flamingos foraging behavior.

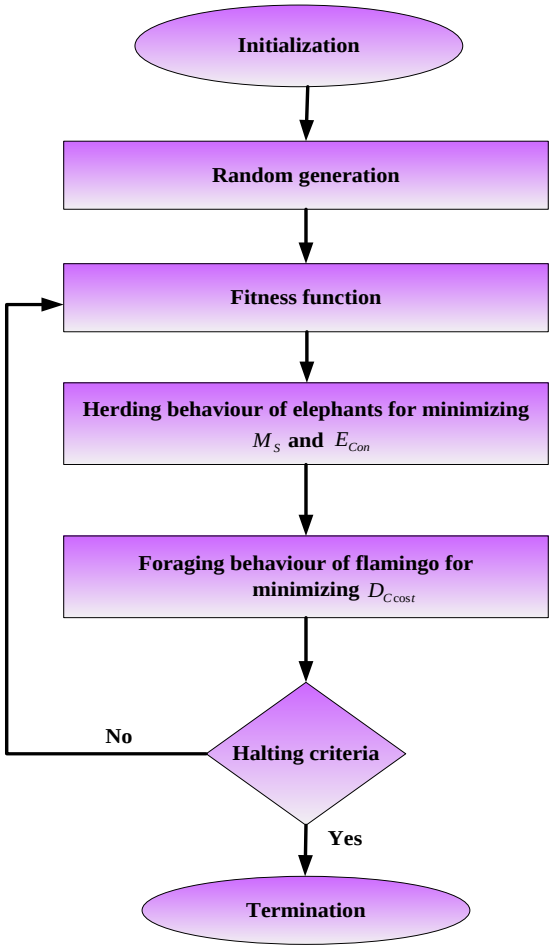

Fig 2: Hybrid elephant herding with Flamingo Search optimization Algorithm to optimize (M_S), (E_{Con}) and ($D_{C\cos t}$)

Step 6: Termination

In this step, the optimal makespan (M_S), energy consumption (E_{Con}) and cost of data center ($D_{C\cos t}$) values are repeated the step three, until the halting criteria is met to load balance in the cloud computing**.**

4. Result and Discussion

The section describes the experimental results for hybrid EHO and FSO based efficient load balancing in the cloud environment. The simulations are done in CloudSiM API 3.0.3. The evaluation matrices, like Mean Response time under various counts of tasks, mean response time under various executable instruction lengths, mean execution time under different count of tasks, count of migrated tasks under increasing count of virtual machines and count of migrated tasks under increasing count of tasks are analyzed to validate the performance of the proposed method. The simulation setup parameters are considered for implementing the proposed LBS-CE-Hyb-EHO-FSA method is tabulated in Table 1.

Table 1: Simulation Parameters

Category	Parameter	Cost
Cloudlets	Count of cloudlets	100-1000
	Task Distance	2000-20000
Data center	Virtual machine scheduler	Time-Shared
	Count of hosts	2-10
	Count of data centres	20
	Cloudlet Scheduler	Time-Shared
	Bandwidth	500-1200
Virtual Machine (VM)	Required number of processor elements	1-4
	Processor Speed	4000-8000 MIPS
	Number of virtual machines	50
	Memory space available in each virtual machine	256-2018 Mb

4.1 Performance Evaluation

The evaluation metrics like mean response time, mean execution time and makespan are considered.

4.1.1 Mean Response Time

Mean response time is a metric used to measure the performance of computer systems and networks. It refers to the average time it takes for a system to respond to a request or task. The response time is calculated as the time elapsed between the initiation of a request and the receipt of a response. Mean response time is obtained by calculating the average of response times over a period of time or a set of requests. It is expressed in equation (25),

$$MRT = \sum_{k=1,1\le s\le n}^{v} RT_s(R_k)/v \quad (25)$$

where RT_s represents the service time of a request, R_k denotes the response time of request and v denotes the service time of the task.

4.1.2 Mean Execution Time

Mean execution time is a metric used to measure the performance of computer programs and algorithms. It refers to the average time taken by a program or algorithm to complete its execution over a period of time or a set of inputs. It is measured using the below equation (26),

$$MET = \left(\frac{Total\ Execution\ Time}{Number\ of\ tasks\ executed} \right) \quad (26)$$

4.1.3 Makespan

Makespan refers to the amount of time required to complete a set of tasks or jobs on a given machines. It is the duration between the start of the first task and the completion of the last task. In other words, makespan represents the total time taken by a system to process a set of jobs from start to finish. Minimizing the makespan is often a key objective in scheduling problems as it can help to optimize resource utilization and improve overall system efficiency and it is calculated using the below equation (27),

$$Makespan = \left(\frac{\sum Max(CT)}{N} \right) \quad (27)$$

here CT specifies the task completion time, N specifies the count of VMs.

Figure 3-12 shows the performance analysis of proposed LBS-CE-Hyb-EHO-FSA method is compared with existing methods, like load balancing optimization based on hybrid heuristic-metaheuristic techniques in cloud environment (LBS-CE-PEFT-ACO) [13], an energy efficient load balancing on cloud computing using adaptive cat swarm optimization (LBS-CE-ACSO) [14], hybridization of firefly and improved multi-objective particle swarm optimization for energy efficient load balancing in cloud computing (LBS-CE-FIMPSO) [15] and adaptive regressive holt–winters workload prediction with firefly optimized lottery scheduling for load balancing in cloud environment(LBS-CE-NMT-FOLS) [16], Machine learning model design for high performance cloud computing & load balancing resiliency: An innovative approach (LBS-CE-XGB) [18] and intelligent Decision-Making of Load Balancing Using Deep Reinforcement Learning and Parallel PSO in Cloud Environment (LBS-CE-DRL- PPSO) [19].

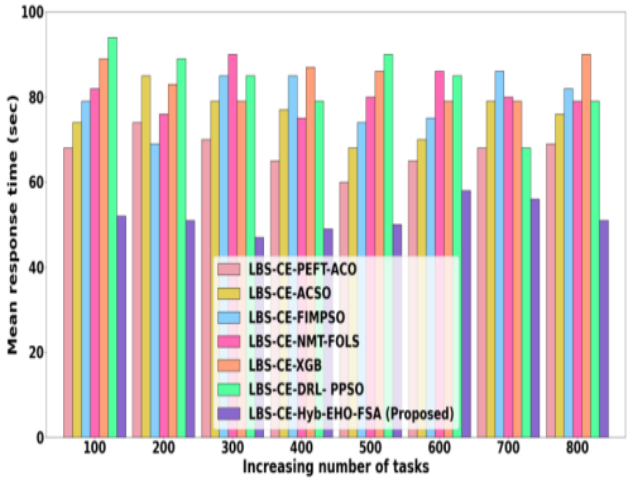

Fig 3: Performance of Mean Response time under different count of tasks

Figure 3 shows the performance of mean response time under various counts of tasks. Here the proposed LBS-CE-Hyb-EHO-FSA method provides 12.08%, 11.03%, 22.06% 10.02%, 3.62% and 15.09% lower mean response time for number of tasks 100; 33.15%, 41.03%, 32.06%, 11.02%, 6.09% and 17.05% lower mean response time for number of tasks 300; 24.30%, 32.10%, 33.12%, 21.02%, 19.27% and 31.12% lower mean response time for number of tasks 500; 33.15%, 21.08%, 6.07%, 11.95%, 15.31% and 22.13% lower mean response time for number of tasks 700 compared with existing methods like LBS-CE-PEFT-ACO, LBS-CE-ACSO, LBS-CE-FIMPSO, LBS-CE-NMT-FOLS, LBS-CE-XGB and LBS-CE-DRL- PPSO respectively.

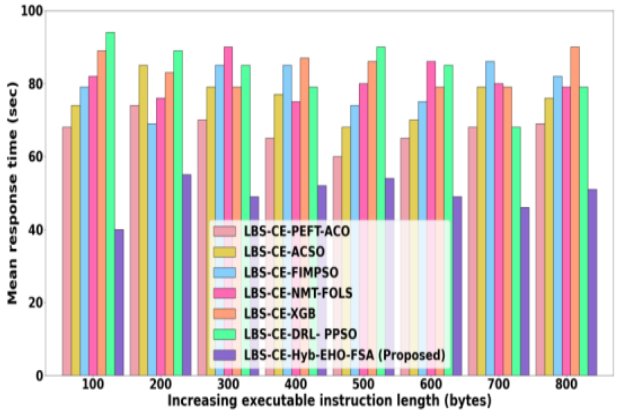

Fig 4: Performance of Mean Response time under different executable instruction lengths

Figure 4 shows the Performance of Mean Response time under different executable instruction lengths Here the proposed LBS-CE-Hyb-EHO-FSA method provides 43.65%, 38.97%, 38.97%, 33.12%, 12.62% and 24.09% lower mean response time for instruction length 0.2; 46.86%, 38.97%, 40.12%, 38.97%, 5.09% and 29.05% lower mean response time for instruction length 0.6; 56.86%, 53.86%, 42.56%, 32.86%, 26.72% and 39.12% lower mean response time for instruction length 1; 34.75%,

56.86%, 34.32%, 37.75% 15.31% and 22.13% lower mean response time for instruction length 1.4 comparing to the existing LBS-CE-PEFT-ACO, LBS-CE-ACSO, LBS-CE-FIMPSO, LBS-CE-NMT-FOLS, LBS-CE-XGB and LBS-CE-DRL- PPSO models respectively.

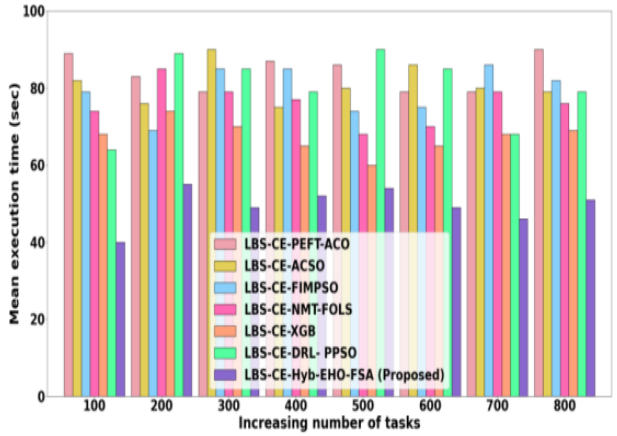

Fig 5: Performance of Mean Execution Time under different count of tasks

Figure 5 shows the performance metrics of mean execution time under various counts of tasks. Here the proposed LBS-CE-Hyb-EHO-FSA method provides 54.75%, 37.86%, 12.32%, 43.76%, 22.62% and 34.09% lower mean execution time for number of tasks 100; 37.76%, 46.87%, 45.12%, 49.86% 21.22% and 19.21% lower mean response time for number of tasks 300; 24.30%, 32.10%, 33.12%, 21.02%, 19.27% and 31.12% lower mean response time for number of tasks 500; 44.65%, 38.65%, 34.87%, 36.86%, 32.15% and 21.31% lower mean response time for number of tasks 700 comparing to the existing LBS-CE-PEFT-ACO, LBS-CE-ACSO, LBS-CE-FIMPSO, LBS-CE-NMT-FOLS, LBS-CE-XGB and LBS-CE-DRL-PPSO models respectively.

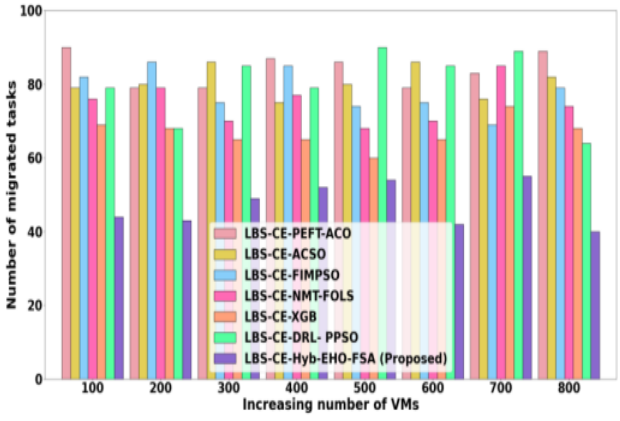

Fig 6: Count of migrated tasks under increasing count of virtual machines (number of tasks=200)

Figure 6 shows the performance metrics of Count of migrated tasks under increasing count of virtual machines (number of tasks=200). Here the proposed LBS-CE-Hyb-EHO-FSA method provides 37.84%, 36.86%, 45.86%, 36.86%, 12.09% and 13.21% lower number of migrated tasks for increasing count of virtual machine 3; 39.12%, 38.75%, 43.86%, 37.86%, 12.22% and 35.19% lower number of migrated tasks for increasing count of virtual machine 5; 24.30%, 32.10%, 33.12%, 21.02%, 19.27% and 31.12% lower number of migrated tasks for increasing count of virtual machine 7; 39.12%, 38.75%, 43.86%, 37.86%, 24.07% and 11.25% lower number of migrated tasks for increasing count of virtual machine 8 comparing to the existing LBS-CE-PEFT-ACO, LBS-CE-ACSO, LBS-CE-FIMPSO, LBS-CE-NMT-FOLS, LBS-CE-XGB and LBS-CE-DRL- PPSO models respectively.

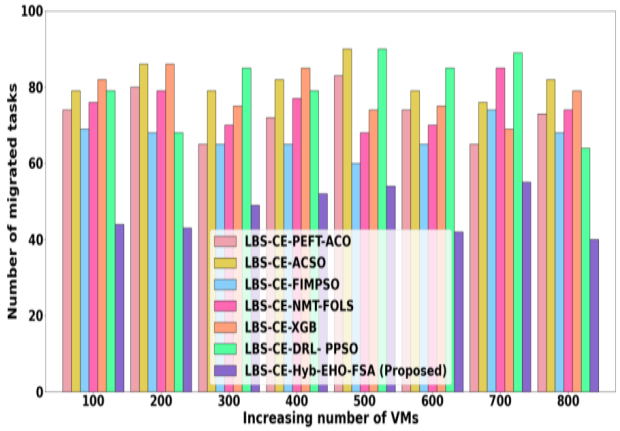

Fig 7: Count of migrated tasks under maximizing count of virtual machines (number of tasks=400)

Figure 7 shows the performance metrics of Count of migrated tasks under increasing count of virtual machines (number of tasks=400). Here the proposed LBS-CE-Hyb-EHO-FSA method provides 43.64%, 36.86%, 45.75%, 39.75%, 21.32% and 18.74% lower number of migrated tasks for increasing count of virtual machine 3; 56.67%, 53.75%, 48.97%, 35.86%, 18.37% and 29.07% lower number of migrated tasks for increasing count of virtual machine 5; 56.34%, 37.97%, 54.86%, 46.86%, 59.01% and 18.79% lower number of migrated tasks for increasing count of virtual machine 7 compared with existing methods like LBS-CE-PEFT-ACO, LBS-CE-ACSO, LBS-CE-FIMPSO, LBS-CE-NMT-FOLS, LBS-CE-XGB and LBS-CE-DRL- PPSO respectively.

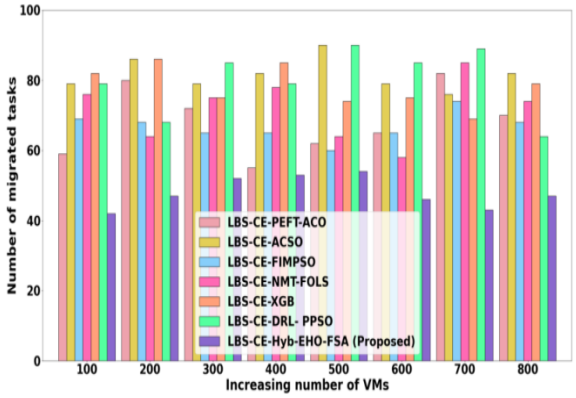

Fig 8: Count of migrated tasks under maximizing count of VMs (number of tasks=600)

Figure 8 shows the performance metrics of Count of migrated tasks under increasing count of virtual machines (number of tasks=600). Here the proposed LBS-CE-Hyb-EHO-FSA method provides 20.56%, 25.74%, 32.32%, 36.75%, 19.57% and 25.37% lower number of migrated tasks for increasing count of virtual machine 3; 26.86%, 31.23%, 43.86%, 42.86%, 28.07% and 9.07% lower number of migrated tasks for increasing count of virtual machine 5; 21.38%, 32.75%, 29.07%, 31.96%, 31.89% and 27.51% lower number of migrated tasks for increasing count of virtual machine 7 comparing to the existing LBS-CE-PEFT-ACO, LBS-CE-ACSO, LBS-CE-FIMPSO, LBS-CE-NMT-FOLS, LBS-CE-XGB and LBS-CE-DRL-PPSO models respectively.

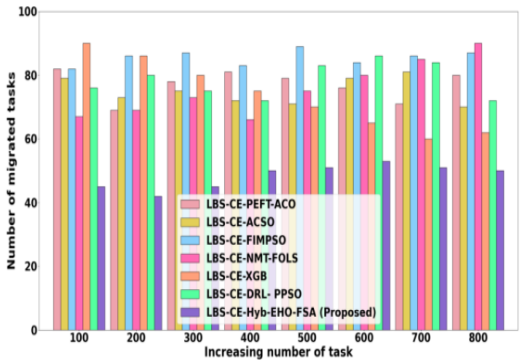

Fig 9: Count of migrated tasks under maximizing the count of tasks (count of virtual machine=2)

Figure 9 shows the performance metrics of Count of migrated tasks under maximizing the count of tasks (count of virtual machine=2). Here the proposed LBS-CE-Hyb-EHO-FSA method provides 45.64%, 37.86%, 37.86%, 36.86%, 15.24% and 24.30% lower count of migrated tasks for increasing count of tasks 200; 26.74%, 32.75%, 37.86%, 36.21%, 14.27% and 51.13% lower number of migrated tasks for increasing count of tasks 600; 49.97%, 48.75%, 27.75%, 39.97%, 36.07% and 34.22% lower number of migrated tasks for increasing count of tasks 800 comparing to the existing LBS-CE-PEFT-ACO, LBS-CE-ACSO, LBS-CE-FIMPSO, LBS-CE-NMT-FOLS, LBS-CE-XGB and LBS-CE-DRL- PPSO models respectively.

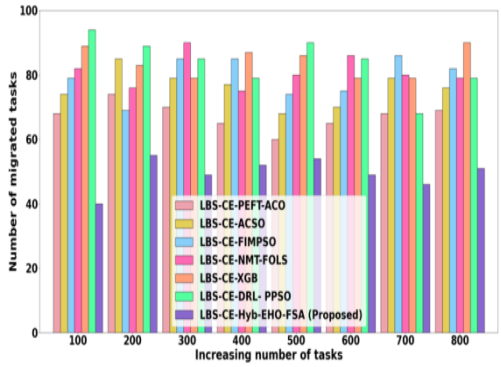

Fig 10: Count of migrated tasks under increasing count of tasks (number of virtual machine=4)

Figure 10 shows the performance metrics of Count of migrated tasks under maximizing the count of tasks (count of virtual machine=4). Here the proposed LBS-CE-Hyb-EHO-FSA method provides 13.55%, 17.98%, 5.14%, 11.17%, 23.34% and 19.27% lower count of migrated tasks for increasing count of tasks 200; 4.62%, 19.57%, 16.58%, 35.98%, 21.34% and 43.21% lower number of migrated tasks for increasing count of tasks 600; 26.526%, 38.458%, 19.52%, 35.45%, 29.34% and 15.47% lower number of migrated tasks for increasing count of tasks 800 comparing to the existing LBS-CE-PEFT-ACO, LBS-CE-ACSO, LBS-CE-FIMPSO, LBS-CE-NMT-FOLS, LBS-CE-XGB and LBS-CE-DRL- PPSO models respectively.

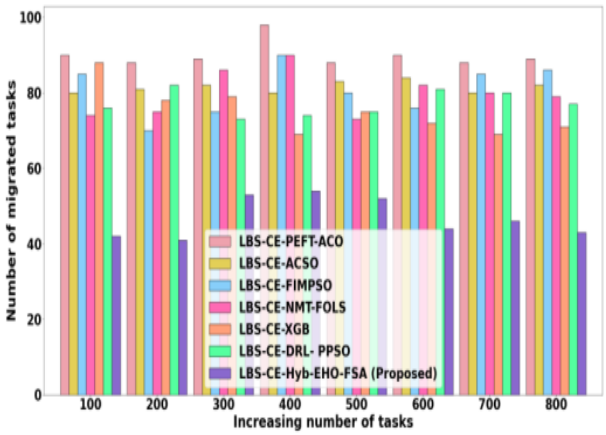

Fig 11: Count of migrated tasks under maximizing count of tasks (count of virtual machine=6)

Figure 11 shows the performance metrics of Count of migrated tasks under maximizing the count of tasks (count of virtual machine=6). Here the proposed LBS-CE-Hyb-EHO-FSA method provides 32.14%, 57.77%, 19.354%, 55.357% 23.34% and 19.27% lower count of migrated tasks for increasing count of tasks 200; 58.23%, 78.378%, 52.63%, 69.66%, 19.56% and 29.51% lower number of migrated tasks for increasing count of tasks 600; 25.63%, 69.66%, 31.11%, 19.65%, 19.07% and 15.47% lower number of migrated tasks for increasing count of tasks 800 comparing to the existing LBS-CE-PEFT-ACO, LBS-CE-ACSO, LBS-CE-FIMPSO, LBS-CE-NMT-FOLS, LBS-CE-XGB and LBS-CE-DRL- PPSO models respectively.

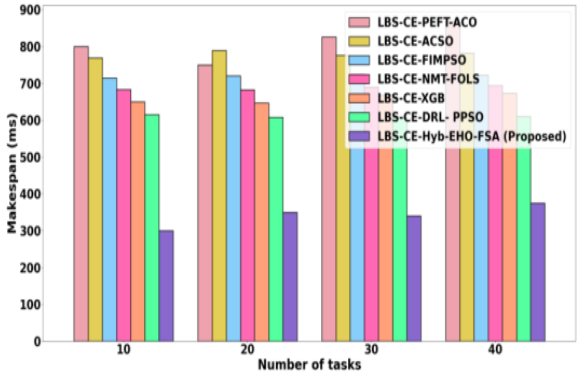

Fig 12: Performance of makespan

Figure 12 shows the performance metrics of makespan. Here the proposed LBS-CE-Hyb-EHO-FSA method provides 41.15%, 22.15%, 60.96%, 15.22%, 24.15% and 14.25% lower makespan for tasks 10; 35.59%, 21.12%, 13.12%, 11.15%, 32.15% and 19.24% lower makespan for tasks 20; 16.52%, 51.03%, 14.33%, 31.88%, 24.01% and 61.09% lower makespan for tasks 30; 10.12%, 32.03%, 12.96%, 17.05%, 25.54% and 31.33% lower makespan for tasks 40 comparing to the existing LBS-CE-PEFT-ACO, LBS-CE-ACSO, LBS-CE-FIMPSO, LBS-CE-NMT-FOLS, LBS-CE-XGB and LBS-CE-DRL- PPSO models respectively.

5. Conclusion

Hybrid Elephant Herding Optimization and Flamingo Search Algorithm are successfully implemented in this manuscript for efficient Load Balancing in a cloud environment. The simulation results proves that the proposed method performance is better under evaluation metrics, viz MRT under various count of tasks and executable instruction lengths, Mean Execution Time under various count of task, count of migrated tasks with various count of virtual machine, count of migrated tasks with various count of tasks. Here, the performance of proposed method LBS-CE-Hyb-EHO-FSA attains 16.04%, 15.56%, 23.43% and 8.507% low MRT under different count of tasks, 13.99%, 11.09%, 12.78% and 8.47% low MRT under different count of tasks at 200 and 15.93%, 56.34%, 23.12% and 9.42% low scheduling time MRT under different count of tasks at 400 compared to the existing LBS-CE-PEFT-ACO, LBS-CE-ACSO, LBS-CE-FIMPSO and LBS-CE-NMT-FOLS methods.

References

[1] A. Kaur, B. Kaur, and D. Singh, 2019. Meta-heuristic based framework for workflow load balancing in cloud environment. *International Journal of Information Technology*, vol. 11, no.1, pp. 119-125.

[2] S.K. Mishra, B. Sahoo, and P.P. Parida, 2020. Load balancing in cloud computing: a big picture.*Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 2, pp.149-158.

[3] V. Arulkumar, and N. Bhalaji, 2021. Performance analysis of nature inspired load balancing algorithm in cloud environment. *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 3, pp. 3735-3742.

[4] S.M. Ali, N. Kumaran, and G.N. Balaji, 2022. 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. 5, p. 2250042.

[5] S. Govindaraju, W.V.R. Vinisha, F.H. Shajin, and D.A. Sivasakthi, 2022. Intrusion detection framework using auto-metric graph neural network optimized with hybrid woodpecker mating and capuchin search optimization algorithm in IoT network. *Concurrency and Computation: Practice and Experience*, vol. 34, no. 24, p.e7197.

[6] P. Arivubrakan, and K. Ramasubramanian, 2023. Multi-Objective Cluster Head based Energy Aware Routing Protocol using Hybrid Woodpecker and Flamingo Search Optimization Algorithm for Internet of Things Environment. *International Journal of Information Technology & Decision Making*.

[7] S.R. Deshmukh, S.K. Yadav, and D.N., Kyatanvar, 2021. Load balancing in cloud environs: Optimal task scheduling via hybrid algorithm. *International Journal of Modeling, Simulation, and Scientific Computing*, vol. 12, no. 02, p. 2150008.

[8] C.T. Yang, S.T. Chen, J.C. Liu, Y.W. Chan, C.C. Chen, and V.K. Verma, 2019. An energy-efficient cloud system with novel dynamic resource allocation methods. *The Journal of Supercomputing*, vol. *75*, pp. 4408-4429.

[9] V.K. Verma, K. Ntalianis, C.M. Moreno, and C.T. Yang,2019Next-generation Internet of things and cloud security solutions. *International Journal of Distributed Sensor Networks*, vol. 15, no. 3, p. 1550147719835098.

[10] C.T. Yang, C.K. Tsung, N.Y. Yen, and V.K. Verma, 2022. Special Issue on Innovative Applications of Big Data and Cloud Computing. *Applied Sciences*, vol. 12, no. 19, p. 9648.

[11] W. Li, and G.G. Wang, 2022. Elephant herding optimization using dynamic topology and biogeography-based optimization based on learning for numerical optimization. *Engineering with Computers*, vol.*38*,(Suppl 2), pp.1585-1613.

[12] W. Zhiheng, and L. Jianhua, 2021. Flamingo search algorithm: a new swarm intelligence optimization algorithm. *IEEE Access*, vol. *9*, pp. 88564-88582.

[13] A. Kaur, and B. Kaur, 2019. Load balancing optimization based on hybrid Heuristic-Metaheuristic techniques in cloud environment.*Journal of King Saud University-Computer and Information Sciences*.

[14] K. Balaji, P.S. Kiran, and M.S. Kumar, 2021. An energy efficient load balancing on cloud computing using adaptive cat swarm optimization. *Materials Today: Proceedings*.

[15] A.F.S. Devaraj, M. Elhoseny, S. Dhanasekaran, E.L. Lydia, and K. Shankar, 2020. 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, vol. 142, pp. 36-45.

[16] J. Prassanna, and N. Venkataraman, 2021. Adaptive regressive holt–winters workload prediction and firefly optimized lottery scheduling for load balancing in cloud.*Wireless Networks*, vol. 27, no. 8, pp. 5597-5615.

[17] S. Ziyath, and S. Senthilkumar, 2021. MHO: meta heuristic optimization applied task scheduling with load balancing technique for cloud infrastructure services.*Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 6, pp.6629-6638.

[18] N.K. Kamila, J. Frnda, S.K. Pani, R. Das, S.M. Islam, P.K. Bharti, and K. Muduli, 2022. Machine learning model design for high performance cloud computing & load balancing resiliency: An innovative approach. *Journal of King Saud University-Computer and Information Sciences*, *34*(10), pp.9991-10009. Nov..

[19] A. Pradhan, S.K. Bisoy, S. Kautish, M.B Jasser, and A.W. Mohamed, 2022. Intelligent Decision-Making of Load Balancing Using Deep Reinforcement Learning and Parallel PSO in Cloud Environment. *IEEE Access*, vol. *10*, pp.76939-76952.

[20] Perez-Siguas, R. ., Matta-Solis, H. ., Millones-Gomez, S. ., Matta-Perez, H. ., Cruzata-Martinez, A. ., & Meneses-Claudio, B. . (2023). Comparison of Social Skills of Nursing Students from Two Universities of Lima. International Journal on Recent and Innovation Trends in Computing and Communication, 11(2), 14–19. https://doi.org/10.17762/ijritcc.v11i2.6105

[21] Jones, D., Taylor, M., García, L., Rodriguez, A., & Fernández, C. Using Machine Learning to Improve Student Performance in Engineering Programs. Kuwait Journal of Machine Learning, 1(1). Retrieved from http://kuwaitjournals.com/index.php/kjml/article/view/101

[22] Aoudni, Y., Donald, C., Farouk, A., Sahay, K. B., Babu, D. V., Tripathi, V., & Dhabliya, D. (2022). Cloud security based attack detection using transductive learning integrated with hidden markov model. Pattern Recognition Letters, 157, 16-26. doi:10.1016/j.patrec.2022.02.012