ISSN PRINT 2319 1775 Online 2320 7876

Research paper © 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, Iss 10, Dec 2022

AN IMPROVED TASK SCHEDULING FRAMEWORK IN CLOUD COMPUTING USING ADAPTED GENETIC LOAD BALANCING MUTATED BINARY PARTICLE SWARM OPTIMIZATION

Mrs. B. Kalaiselvi and Prof. Mary Immaculate Sheela L
[1]Assistant Professor & HOD, SIVET College, Chennai, Tamil Nadu
[2]Dean, Faculty of Engineering Science and Computing (FESAC), Pentecost University, Accra, GHANA, West Africa

ABSTRACT
In cloud computing environment tasks are allocated among virtual machines (VMs) having different length, starting time and execution time. Therefore, balancing these loads among VM is a key factor. Load balancing has to be carried out in such a manner that all VMs should have balanced to achieve optimal utilization of its capabilities and improve the system performance. In this work, proposed a load balancing and task scheduling technique by using Load Balancing Mutated Binary Particle Swarm Optimization (LBMBPSO) with multi-objective concept to schedule tasks over the available cloud resources that minimizes the makespan and maximizes resource utilization. This is achieved by having proper information among the tasks and resources within the datacentre. This work adopts concepts of the DNA representation and the mutation operator of genetic algorithms. The proposed LBMBPSO algorithm is tested on various benchmark functions, and its performance is compared with that of the original BPSO. The proposed scheduling algorithm is implemented by using CloudSim simulator. Simulation results clearly shows that proposed scheduling algorithm performs better in reducing make span and increases the resource utilization than other existing techniques.

Keywords: *Cloud Computing, Load Balancing, Task Scheduling, Adapted Genetic Binary Particle Swarm Optimization, DNA Representation and Mutation Operator.*

INTRODUCTION

Cloud computing is experiencing a rapid development both in academia and industry; it is promoted by the business rather than academic which determines its focus on user applications. This technology aims to offer distributed, virtualized, and elastic resources as utilities to end users. It has the potential to support full realization of 'computing as a utility' in the near future [1]. With the support of virtualization technology [2], cloud platforms enable enterprises to lease computing power in the form of VM to users. Because these users may use hundreds of thousands of VMs [3], it is difficult to manually assign tasks to computing resources in clouds [4]. These days, cloud computing has become a well-known commercial computing paradigm. It can offer various computing services to users with virtual machine as the resource unit such as storage, applications, networks and servers over the Internet (5).

Users or clients can get these resources on request according to the Service Level Agreement and user pay for which they consume the services for specific duration of time [6]. Cloud resources having different parameters such as storage, memory, network and processing speed which can be given to the user as-a-service. In cloud environment, each data center consist of number of host and each physical host can stack at least one virtual machine with the goal that clients can run the applications freely [7]. It transforms into a troublesome activity to manage the entire request from customer side in the most restricted reaction time and satisfied nature of administration. Responsibility of cloud service provider (CSP) is to use these resources appropriately and keep the load as balanced in between these resources. Resources should be assigned to incoming request depending upon the property of resource information, task information and proper scheduling algorithm [8].

Load balancing is a challenging concept in cloud computing [9] is difficult to arrange these resources in the cloud because the workload in the cloud may fluctuate from time to time as per user requirement [10]. Load balancing is a process to distribute the request between different machines through task scheduling so that numbers of jobs are executing with less time and monitors the performance of VMs. The central aim of load balancing technique is to reduce makespan time while increasing the usage of resources of cloud [11].

In task scheduling the virtualized resources can be assigned to the particular task for specified amount of time. It can be done by using task scheduling algorithm which is to be handled by cloud resource broker. Task scheduling shows the upcoming task will be executed in least time. CSP gets the status of running virtual machine constantly so as to discover a superior resource for upcoming task. After that it performs load balancing operation to maintain the load of all VMs. Task scheduling is an NP-hard combinational optimization problem in cloud environment because of number of tasks increases and length of task changes quickly. It is difficult to establish the mappings between task and resources. Hence, need a proficient task scheduling technique which can better deal with the task and tackle NP-hard issue. For such issue, many researchers focused their researching work on heuristic, meta-heuristic and hybrid scheduling strategies [12]. At present, the swarm intelligence algorithms are well used for resolving these kinds of problems. among various meta-heuristics algorithms, PSO is a famous metaheuristic technique to solved optimization issue that is appropriate for dynamic task scheduling, workflow scheduling and load balancing. PSO has a strong worldwide searching capability toward the start of the run and a nearby pursuit close to the furthest limit of the run. Therefore, it has been generally utilized in different applications and has made incredible progress [13]. But when the problem continues to expand then past existing PSO algorithm is not an effective technique for all scenarios.

In this work, a modified Binary PSO algorithm named as LBMBPSO is proposed to solve the problem of load balancing and task scheduling. LBMBPSO task scheduling method is based on BPSO algorithm which utilizes a fitness function to evaluate the ideal arrangement of every particle. The fitness function is computing the execution times of each VM and return the most elevated execution time as the fitness value of each PSO particle. Originally, the position was updated by combining its current position and velocity, but in BPSO, the position is updated by reflecting only the current velocity; generally, the sigmoid function has been used to update the position in BPSO. Due to this characteristic, it seems that the velocity domain is a search space though an actual binary search space already exists. Based on genetics, any organism can be represented by its phenotype, which virtually determines what exactly the object is in the real world, and its genotype containing all the information about the object at the chromosome set level. The role of each gene is reflected in the phenotype. In this work, LBMBPSO using this DNA paradigm of genetics demonstrated will match the genotype and the phenotype to the velocity and the binary position, respectively, in BPSO.

The remainder of this paper is arranged as follows: Section 2 shows the related work and Section 3 depicts the problem formulation. In Section 4 we describe the standard PSO technique. Section 5 describes model structure of LBMPSO. Section 6 shows proposed technique. Section 7 shows the simulation tool and experimental results. The conclusion and future work of the paper is given in Section 8.

Related Work

Haris & Zubair [14] proposed a dynamic load balancing algorithm based on the hybrid optimization algorithms named as Mantaray modified multi-objective Harris hawk optimization (MMHHO). The hybridization process updates the search space of Harris Hawk Optimization (HHO) by utilizing the Manta Ray Forging Optimization (MRFO) algorithm by considering the cost, response time, and resource utilization etc. The hybrid scheme, proposed in the present study, improves the system performance by enhancing the VMs throughput, balancing the load between the VMs, and sustaining the balance among priorities of tasks by adjusting the waiting time of the involved tasks. However, most scheduling algorithms customarily lead to less resource utilization, termed load imbalance.

Narwal & Dhingra [15] rendered Credit-based Resource Aware Load Balancing scheduling algorithm (CB-RALB-SA). The proposed work ensures a balanced distribution of tasks based on the capabilities of the resources, which eventually proves sustainable improvement against the existing scheduling algorithms. The tasks weighted by the credit-based scheduling algorithm are then mapped to the resources considering each resource's load and computing capability using FILL and SPILL functions of Resource Aware and Load using Honey bee optimization heuristic algorithm. Thus, it improves the processor's efficiency while uplifting the whole system's performance and has saved memory allocated to tasks and RAM. However, these systems were unable to take into account the resource's ability to do a task as well as the user's requirements.

Zade & Mansouri [16] presented to enhance the ability of the Red Fox Optimization (RFO) algorithm, a Quasi-Opposition Based Learning method is employed for generating the initial population and a Levy flight method is used to enhance the exploration ability of newly generated foxes. In addition, an efficient task scheduling using the Fuzzy Improved RFO (FIRFO) algorithm and game theory named EGFIRFO is presented considering fthe conflicting objectives (i.e., resource utilization, load balancing, makespan, and execution time). As a global optimization, the proposed method is compared with Bat algorithm (BA), Grey Wolf Optimizer (GWO), Particle Swarm Optimization (PSO), Antlion Optimizer (ALO), and Red Fox Optimizer (RFO) in terms of a set of qualitative parameters (i.e., convergence curve, trajectory, average fitness, and search history; however, if the loads fluctuate randomly throughout the duration of execution time, it will not operate correctly.

Manikandan et al., [17] proposed hybrid WOA based MBA algorithm, the multi-objective behavior decreases the makespan by maximizing the resource utilization. The output of the Random double adaptive whale optimization algorithm (RDWOA) is enhanced by utilizing the mutation operator of the Bees algorithm. The performance evaluation is conducted and compared with other algorithms using the platform of Cloudsim tool kit for various measures such as completion, time, and computational cost. The proposed HWOA based MBA algorithm converged faster than any other approach for large search spaces and makes it appropriate for large scheduling problems. The experimental results reveal that the HWOA based MBA algorithm effectively minimizes the task completion time and aLBMBPSO execution time. However, this method suffers time-consuming and imbalance between diversity and convergence.

Mapetu et al., [18] proposed an efficient binary version of PSO algorithm with low time complexity and low cost for scheduling and balancing tasks in cloud computing. Specifically, we define an objective function which calculates the maximum completion time difference among heterogeneous VMs subject to updating and optimization constraints introduced in this paper. Then, we devise a particle position updating with respect to load balancing strategy. The experimental results show that the proposed algorithm achieves task scheduling and load balancing better than existing meta-heuristic and heuristic algorithms. However, these algorithms do not guarantee that the optimal solution can be found, if they are not combined with other heuristic or meta-heuristic algorithms.

Venkataraman [19] proposed Threshold Based Multi-Objective Memetic Optimized Round Robin Scheduling (T-MMORRS) Technique. Finally, the selected load balancing algorithm in MMORRS Technique schedules the user request task to a resource-efficient virtual machine with higher efficiency and lower time consumption. As a result, T-MMORRS Technique enhances the task scheduling performance to balance the both bursty and non-bursty workloads of VM in the cloud. The experimental evaluation of T-MMORRS Technique is conducted using factors such as scheduling efficiency, scheduling time and energy consumption with respect to the number of user requests. The experimental result shows that the T-MMORRS Technique can enhance the scheduling efficiency and LBMBPSO minimizes the energy usage in the cloud as compared to state-of-the-art works. However, scheduling performance of existing technique was not effective in burstiness workload's conditions.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper © 2012 IJFANS. All Rights Reserved, UGC CARE Listed (Group -I) Journal Volume 11, Iss 10, Dec 2022

Thus, there is a need for a novel task scheduling technique to handle bursty user demands and provide high-quality cloud services.

Prassanna & Venkataraman [20] VM consolidation technique called Nature-inspired Meta-heuristic Threshold based firefly optimized lottery scheduling (NMT-FOLS) Technique is proposed. NMT-FOLS Technique applies multi-objective firefly optimization-based task scheduling algorithm in normal workload state and multi-objective firefly optimized lottery scheduling algorithm in timely and bursty workload situations. At last, the selected scheduling algorithm in NMT-FOLS Technique assigns the user requested task to best VMs in CS to perform the demanded services. From the experimental result, the NMT-FOLS technique improves scheduling efficiency up to 94.6% and reduces the SLA violations and energy utilization from different test cases on an average to 78%, and 63% compared to state-of-the-art works. However, scheduling performance of existing technique was not effective in burstiness workload's conditions.

Nabi et al., [21] proposed a resource-aware dynamic task scheduling approach and the simulation experiments have been performed on the Cloudsim simulation tool considering three renowned datasets, namely HCSP, GoCJ, and Synthetic workload. The obtained results of the proposed approach are then compared against RALBA, Dynamic MaxMin, DLBA, and PSSELB scheduling approaches concerning average resource utilization (ARUR), Makespan, Throughput, and average response time (ART). The DRALBA approach has revealed significant improvements in terms of attained ARUR, Throughput, and Makespan. This fact is endorsed by the average resource utilization results (i.e., 98 % for HCSP dataset, 75 % for Synthetic workload (improve ARUR by 72.00 %, 77.33 %, 78.67 %, and 13.33 % as compared to RALBA, Dynamic MaxMin, DLBA and PSSELB respectively), and 77 % for GoCJ (i.e., the second best attained ARUR)). To reserve more resources can enhance the performance of the user application requirements; however, it will increase the resource usage cost.

Kruekaew & Kimpan [22] proposed an independent task scheduling approach in cloud computing using a multi-objective task scheduling optimization based on the Artificial Bee Colony Algorithm (ABC) with a Q-learning algorithm, which is a reinforcement learning technique that helps the ABC algorithm work faster, called the MOABCQ method. The proposed method aims to optimize scheduling and resource utilization, maximize VM throughput, and create load balancing between VMs based on makespan, cost, and resource utilization, which are limitations of concurrent considerations. However, tasks should be distributed among all VMs in parallel to balance the system and ensure efficient use of available resources.

Ali et al., [23] suggested an optimization model based on a Discrete Non-dominated Sorting Genetic Algorithm II (DNSGA-II) to deal with the discrete multi-objective task-scheduling problem and to automatically allocate tasks that should be executed either on fog or cloud nodes. The NSGA-II algorithm is adapted to discretize crossover and mutation evolutionary operators, rather than using continuous operators that require high computational resources and not able to allocate proper computing nodes. In the model, the communications between the fog and cloud tiers are formulated as a multi-objective function to optimize the execution of tasks. The proposed model allocates computing resources that would effectively run on either the fog or cloud nodes. Moreover, it efficiently organizes the distribution of workloads through various computing resources at the fog. Several experiments are conducted to determine the performance of the proposed model compared with a continuous NSGA-II (CNSGA-II) algorithm and fthe peer mechanisms. The outcomes demonstrate that the model is capable of achieving dynamic task scheduling with minimizing the total execution times (i.e., makespans) and costs in fog-cloud environments. There is a need to have intelligent resource management schemes to satisfy the applications' performance requirements and efficiently utilize the available computing resources.

Pang et al., [24] developed an EDA-GA hybrid scheduling algorithm based on EDA (estimation of distribution algorithm) and GA (genetic algorithm). First, the probability model and sampling method of EDA are used to generate a certain scale of feasible solutions. Second, the crossover and mutation operations of GA are used to expand the search range of solutions. Finally, the optimal scheduling strategy for assigning tasks to VM is realized. This algorithm has advantages of fast convergence speed and strong search ability. The experimental results show that the EDA-GA hybrid algorithm can effectively reduce the task completion time and improve the load balancing ability. However, there were some problems with load imbalance and reducing resource utilization.

Inference: Most of the existing methods do not consider load balancing factors when attempting to reduce the task completion time. For instance, to reduce task completion time, it is easy to centrally schedule the tasks on the resources with strong computing power, which will cause a load imbalance problem. Therefore, it is challenging to design and optimize the task scheduling algorithm to balance the two goals of reducing completion time and improving load balancing ability. In this work, hybrid algorithm is proposed to solve the multi-objective task scheduling problem with the criteria of reducing task completion time and improving load balancing ability.

PROPOSED METHODOLOGY

Fig.1 illustrates the system architecture of LBMBPSO based task scheduling. The consumers who wish to use cloud computing services will submit their service requests to the cloud provider. Then the cloud provider needs to find an optimized schedule solution for the submitted workflow application request. This optimized schedule solution represents the optimal task execution VMs denoted as nodes that will minimize the make span of the system. The next section presents the performance of the proposed approaches with LBMBPSO method on solving the task scheduling problems in the cloud computing environment.

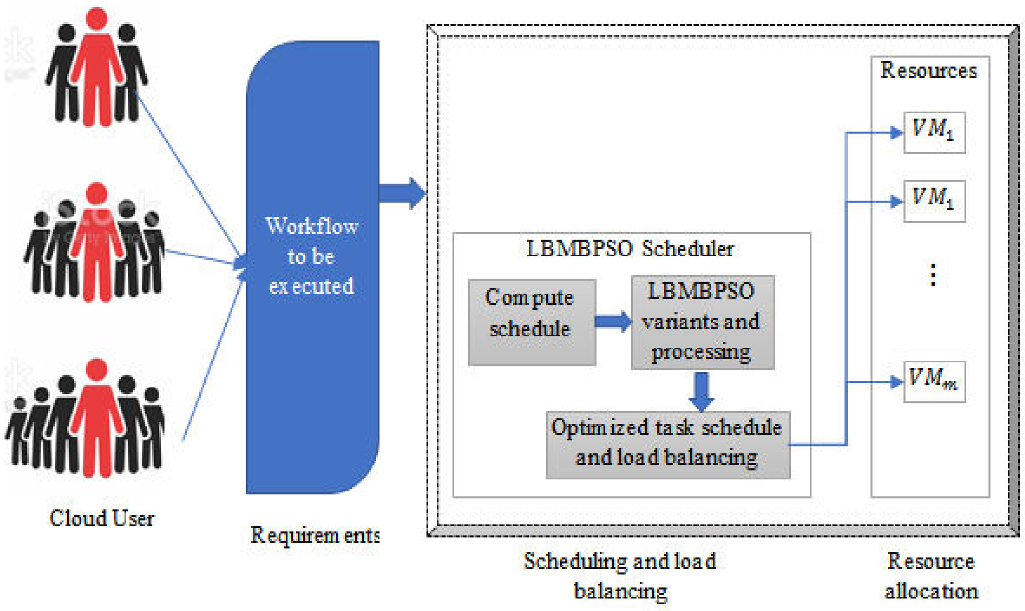

Fig.1. System architecture of LBMBPSO-based scheduler

Problem Statement

Consider swarm optimization with p particles, m VMs of heterogeneous types, and n distinct forms of workload ($\mathcal{W}$). This position allocation matrix $\mathcal{PAM}$, expressed as $(m \times n) = (m + n)$, depicts the distribution of jobs across these VM. This is a position allocation matrix for a particle k that specifies where the task corresponds to $\mathcal{PAM}^k$ as in Eq.(1).

$$\mathcal{PAM}^{k} = \begin{array}{c} VM_1 \\ VM_1 \\ \vdots \\ VM_m \end{array} \begin{bmatrix} x_{1,1} & \cdots & x_{1,n-1} & x_{1,n} \\ x_{2,1} & \cdots & x_{2,n-1} & x_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m,1} & \cdots & x_{m,n-1} & x_{m,n} \end{bmatrix}, x_{ij} = \begin{cases} 1 & if\ \mathcal{W}_j\ is\ assigned\ to\ VM_i \\ 0 & otherwise \end{cases} \quad (1)$$

Each particle in the swarm represents a possible solution that could lead to a viable solution. In other words, this indicates that the perfect solution may exist within the subatomic particle k. Therefore, in this procedure, all particles are subjected to the same number of iterations prior to the identification of a particle that provides the expected ideal response, based on repeated comparisons between these particles. As illustrated by the equation, one technique to expedite the procedure is to iterate each particle the same number of times. As a meta-heuristic algorithm, BPSO becomes less applicable in real-world circumstances [25]. If you lack a particle that can produce an optimum response, you cannot guarantee an ideal outcome. Here, elaborate on a new, low-complexity, low-cost LBMBPSO for scheduling and balancing heterogeneous cloud-based virtual machine operations. First, we explain the suggested LBMBPSO framework and define the problem-related objective function in order to be more specific. To accelerate research into binary space, have established a revolutionary concept and calculation for each particle in the model. This provides for substantial time savings while delivering the highest quality solution available. In light of this, present two constraints, namely update and optimization constraints, to determine how many fitness solutions can quickly generate an ideal solution. The final step in the effort to reduce simulation costs is to develop a more accurate model of particle position, based on a load balancing method with an updating constraint. The fitness evaluation technique has been improved to prevent particles from wandering too far from the optimal response.

BPSO can be utilised to address the problem of job scheduling and load balancing utilising the cloud computing paradigm depicted in Fig.1. These are the three components that comprise the cloud system; the demands of users are divided into several tasks and sent to the cloud management as the initial module. For each virtual machine, cloud management (CM) generates a local work queue (VM). LBMBPSO, together with the pricing model and mapping, is a submodule of CM. LBMBPSO schedules all jobs across heterogeneous VMs based on updating and optimization constraints [54–59]. Mapping assigns each local queue to a VM, and the price model determines the execution cost for all user tasks. A virtual machine manager for many hosts concludes the third module. Numerous VM are deployed for various purposes. In contrast to other PSO algorithm, the LBMBPSO algorithm inherits the basic ideas from BPSO algorithm with Geno-phenotype concept to decrease the computation time of tasks executing, it also considers the loading of each VM that can carry out new task scheduling depending on the result in the past task scheduling. It is very helpful in the cloud environment.

Multi-Objective Function: The problem examined in this paper is related to the identification of the optimal configuration of virtual machine placement within the servers hosted in different datacenters in such a way to optimize multiple objectives including, Completion time, Qulaity of Connection, and load balancing. VMs running on a variety of hosts and performing a range of tasks each have their own unique completion time $\mathcal{CT}$, according to Eq.(2). The completion time difference of m diverse loads is defined as:

$$\mathcal{CT} = \left| \frac{\sum_{i=1}^{n} \mathfrak{LT}_i}{\sum_{i=1}^{n} \sum_{j=1}^{m} \mathfrak{PS}_{ij}} \right| \quad (2)$$

$$\mathfrak{LT}_i = \sum_{i=1}^{n} \frac{length}{VM_i{\cdot}PEs \times VM_i{\cdot}MIPS} \quad (3)$$

The processing speed $\mathfrak{PS}_{ij}$related to $\mathfrak{LT}_i$running on VM_jin the cloud depends on how many request tasks have been mapped to that VM (or n) as well as the total allocated MIPS of VM_jalong all its processing elements or $\mathfrak{PS}$ (or Capacity ca_j). Calculation of the request processing speed $\mathfrak{PS}$ is shown in Eq.(4):

$$\mathfrak{PS}_{ij} = \frac{ca_j}{n} \tag{4}$$

Where length (in millimetres imperial), the ID of a virtual machine is its number of processing elements (PEs), the MIPS is its execution speed per PE, and the prenumber is the number of PEs within the virtual machine, i and j have values between 1 and m, and i and j are not equivalent. $\mathcal{CT}$ is meant to identify VM that are overcrowded or underloaded. As indicated in Equation (2), the objective function attempts to reduce the disparity between available heterogeneous VMs and overall completion time, while reducing user task waiting times. This strategy simultaneously reduces time and metrics. As a result, Eq. (5) represents the objective function OF:

$$OF(\mathcal{CT}) = \max\left(\frac{\mathcal{CT}}{1\leq i \mid j\leq m}\right) + \alpha.LB_j + \beta.QoC_j, \alpha,\beta \in 1 \tag{5}$$

LB_j is the load balancing factor of VM_j, to minimize the degree of imbalance, which is defined as follows

$$LB_j = 1 - \frac{E_j - l}{E_j + l}$$

Where l is the average execution time of the virtual machines in the last iteration of the optimal resource allocation, and E_j is the expected execution time of the task in the VM_j, which is defined as $E_j = \frac{T}{I}$, Where T is the total length of the tasks that have been submitted to VM_j, and I is the length of the task before execution. As restrictions for updating and optimising, the sigmoid function and the minimum completion time for unique VMs are utilised, respectively.

Quality of Connection (QoC) the aim is to equalize the average QoC of all VM_j, i.e., the arithmetic mean of the QoC experienced by connections of a cloud service. QoC the overall acceptability of an application or service, as perceived subjectively by the end-user.

$$QoC_j = \frac{1}{N_j}\sum_{i=1}^{N_j} QoC_i$$

where j is the evaluated service, N_j is the number of users of cloud service j, and QoC_i is the QoC perceived by user i, estimated from QoS statistics. The sum considers that all users of the same service have equal target QoC.

LBMBPSO: In this section, LBMBPSO, is described using the DNA concept and the mutation operator. The velocity and the binary position parameters of the original BPSO are corresponding to the genotype of position and phenotype of position of the modified BPSO, respectively. Furthermore, from considering a characteristic of the genotype of position (velocity in the original BPSO), suggest a mutation operator for improving the performance of the BPSO based on the previous work [8].

DNA concept: The main difference between the PSO and the BPSO is in the position update function. Specifically, the position update of the BPSO does not use the information of the current position. In other words, the next position of the BPSO is not influenced by the current position but influenced by the velocity only. This implies that when updating a position in the BPSO, it is meaningless to know where a particle is currently located in the binary search space. Because of this fact, it seems that velocity is a particle, even though the binary position already exists in the BPSO. Thus, we suggest the concept that the velocity and the position of the original BPSO be taken as a particle and a solution transformed by the sigmoid function, respectively. This concept is based on the imitation of the mechanism of DNA concept in biology. The genotype of an individual is the genetic information carried by the individual's genes, whether or not the genes are expressed. The phenotype denotes all the observable characteristics of an individual, such as physical appearance (eye color, height, etc.) and internal physiology. It is determined partly by genes, the environment and the way of life.

In genetic algorithms, the DNA concept has already been applied to encoding schemes. As an example, we introduce random key encoding [9]; generally, random key encoding is used for an order-based encoding scheme. If there are five genes in a chromosome, each gene is assigned a random number drawn uniformly from (0, 1). To decode the chromosome, count the genes in ascending order with regard to their values as the following for Random key, 0.42, 0.06, 0.38, 0.48,0.81 Decodes as 3,1,2,4,5 respectively, where the value of random key is the genotype and the value of decode is the phenotype. Note that genes to be counted early tend to "evolve" closer to 0, and those to be counted later tend to evolve closer to 1 because of the ascending order.

The DNA concept can be applied to the BPSO as follows: Let the velocity of the original BPSO be a continuous search space, and then a binary position can be decoded by the sigmoid function. Here, let the velocity and the binary position of the original BPSO be a genotype $x_{\mathrm{g},i,j}$ and a phenotype $x_{\mathrm{p},i,j}$, respectively. Then, the update functions of the original BPSO are changed as the following expressions. In the velocity update Eq. (6), the acceleration coefficients, $\mathfrak{ac}_1$ and $\mathfrak{ac}_2$ (must be greater than 1) determine the influence of the personal best and the neighborhood best solutions on the particle's current velocity vector. In the original BPSO each particle's velocity $\mathbb{v}$ and position x are modified by the following formula:

$$\mathbb{v}_{i,j}(\mathfrak{g}+1) = \mathfrak{iw} \cdot v_{i,j}(\mathfrak{g}) + \mathfrak{ac}_1\mathbb{R}_1\left(p_{best,i,j} - x_{p,i,j}(\mathfrak{g})\right) + \mathfrak{ac}_2\mathbb{R}_2\left(g_{best,i,j} - x_{p,i,j}(\mathfrak{g})\right) \quad (6)$$

$$x_{\mathrm{g},i,j}(\mathfrak{g}+1) = x_{\mathrm{g},i,j}(\mathfrak{g}) + \mathbb{v}_{i,j}(\mathfrak{g}+1) \quad (7)$$

$$x_{\mathrm{p},i,j}(\mathfrak{g}+1) = \begin{cases} 0 & if\ rand \geq \frac{1}{1+e^{-x_{\mathrm{g},i,j}(\mathfrak{g}+1)}} \\ 1 & otherwise \end{cases} \quad (8)$$

Where $\mathbb{R}_1$ and $\mathbb{R}_2$ are random numbers. In update functions of the LBMBPSO, the information concerning the positions used in the velocity update (6) is from the phenotype not from the genotype of the positions. The LBMBPSO is identical to the original BPSO with 1.0 inertia weight if $\mathfrak{iw} = 0$ in the velocity update function (6).

Mutation Strategy: When the original BPSO algorithm starts the iteration to find an optimum, the velocity tends to go into $\mathbb{v}_{max}$ or $-\mathbb{v}_{max}$ by the velocity update if the corresponding target position is one or zero. If a velocity converges to near $\mathbb{v}_{max}$ or $-\mathbb{v}_{max}$, it is hard to change the corresponding position with a small change of velocity, which makes it difficult to escape from a good local optimum in the BPSO. In order to get out of this undesired position, large movement of velocity is required, which is unconcerned with two current best positions (pbest and gbest). For accomplishing the above objective, suggest the insertion of the following operation between velocity update and position update of the LBMBPSO process:

$$for\ (i = 1; i < n; i = i+1)\left\{ if\ (rand < r_{mut})\ then\ \mathbb{v}_{i,j_r}(\mathfrak{g}+1) = -\,\mathbb{v}_{i,j_r}(\mathfrak{g}+1)\right\} \quad (9)$$

$$for\ (i = 1; i < n; i = i+1)\left\{ if\ (rand < r_{mut})\ then x_{\mathrm{g},i,j_r}(\mathfrak{g}+1) = -x_{\mathrm{g},i,j_r}(\mathfrak{g}+1)\right\} \quad (10)$$

where $r_{mut}$is the probability that the proposed operation is conducted in the ith particle and j_r is the randomly-selected position of this particle. If this operation is executed when velocity is near $\mathbb{v}_{max}$ or $-\mathbb{v}_{max}$, the position will be changed from one to zero or from zero to one, respectively. This rule similarly affects the BPSO algorithm with a bit change mutation in GAs [10]; i.e., regarding a certain probability, a bit is changed from 0 to 1 or from 1 to 0, respectively. This procedure of optimized VMs solution is given in Table 1. The Fig. 1. shows the flowchart of LBMBPSO algorithm.

Table 1: Algorithm for LBMBPSO for load balancing and task scheduling

```
• Begin
• $\mathrm{g} = 0$; {t: generation index}
• Initiate the position and velocity vectors of each particle (virtual machine);
• Calculate the matrix and fitness value of each particle using a fitness function.
• This is the first time that the best location has been allocated to a particle's "pbest", use the particle's current position value instead if its current fitness value is higher than its "pbest".
• The particle with the highest fitness value should be chosen as the best.
• Choose VM (best particle) for next task by velocity and position updation.
• Evaluation $x_{\mathrm{p},i,j}$;
• while (termination condition≠true) do
• update $\mathrm{v}_{i,j}(\mathrm{g}+1)$; {by Eq. (6)}
• update $x_{\mathrm{g},i,j}(\mathrm{g}+1)$; {by Eq. (7)}
• mutation$x_{\mathrm{p},i,j_r}(\mathrm{g}+1)$; {by Eq. (9)}
• $x_{\mathrm{p},i,j}(\mathrm{g}+1)$= decode $x_{\mathrm{g},i,j}(\mathrm{g}+1)$;
• evaluate $x_{\mathrm{p},i,j}(\mathrm{g})$;
• $\mathrm{g} = \mathrm{g} + 1$;
• end while
• Once all iterations have been completed, we have reached the terminating condition when there is no further change in particle fitness.
• Finally, the best particle is produced. Sustainability.
• End
```

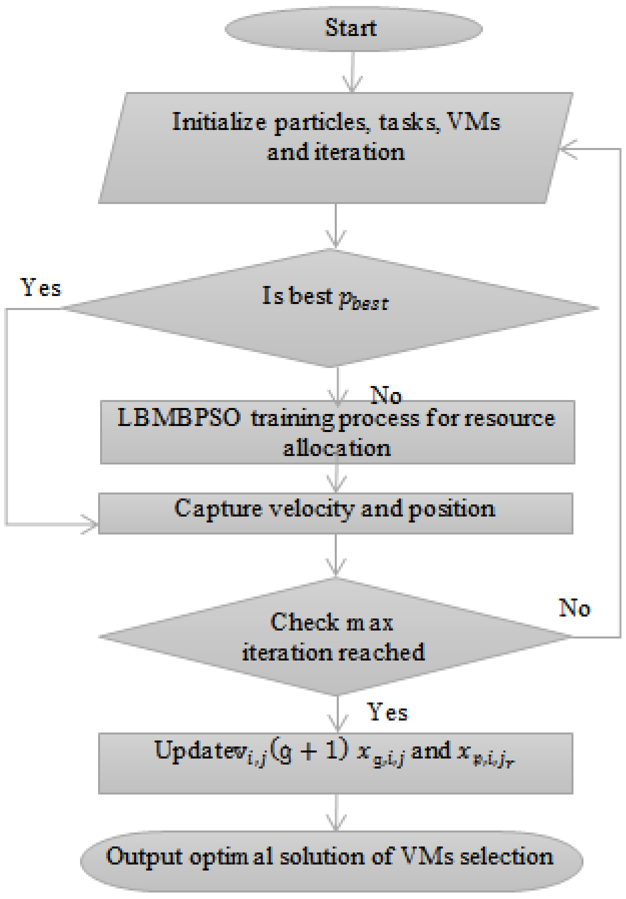
Start
Initialize particles, tasks, VMs and iteration
Is best p_{best}
Yes
No
LBMBPSO training process for resource allocation
Capture velocity and position
Check max iteration reached
No
Yes
Update $v_{i,j}(g+1)$ $x_{g,i,j}$ and x_{p,i,j_r}
Output optimal solution of VMs selection

Fig.2: Flow diagram of LBMBPSO based task scheduling

EXPERIMENTAL RESULTS AND DISCUSSION

In this section, compare the proposed LBMBPSO method to the state-of-the art scheduling algorithms such as MMHHO [14], EDA-GA [24] and ANN-BPSO [25], already in use, and explain the experimental findings and evaluation of its performance. Three other scheduling algorithms were compared to the suggested algorithm to see how well it worked in terms of make-span, average waiting time, response time, degree of imbalance, total cost and resource utilization.

$$resource\ utilization = \frac{\sum_{i=1}^{N} T_{VM_i}}{MS \times N} \tag{11}$$

Where $T_{VM_i}$is the time taken by the VM_i to finish all jobs, MS is the makespan and N is the number of resources

$$Totalcost = \sum CT_{ij} \times C_{VM_j} \tag{12}$$

Where $ET_{ij}$the completion is time of the task i on VM_j and C_{VM_j} is the cost of VM_j per unit time.

Experimental Setup: It is difficult to try out new approaches or ideas when the infrastructure is inflexible, as it often is in a cloud computing environment like Amazon EC2 or Microsoft Azure, because of issues like security, speed, and the high cost in currency of repeating testing [60–64]. These sorts of tests are difficult to execute on real-world cloud infrastructures since they need a lot of effort to make them scalable and repeatable. According to previous research, the CloudSim-3.0.3 simulator may be used to assess a proposed algorithm's performance in real-world scenarios. Additional data was created at random by the simulator itself and used in the testing. It was possible to write all of the Java code and execute it concurrently on an Intel Core i5-6500 PC running at 3.2 GHz with 4GB of RAM. Table 3 lays out the components and parameters of the model. First, the experiment on independent jobs and the impact of update coefficient, and second, experiments on varied workloads and number of virtual machines are related to these factors in CloudSim parameters. Since all VMs are spread evenly across all hosts, the first value represents how things were originally set up. The VMs are distributed unevenly between hosts, which is reflected in the second result.

Makespan Comparison Results

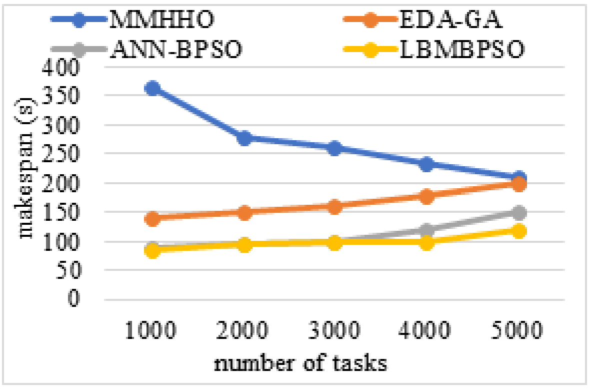

Fig.3. make span comparison results

In terms of makespan, the suggested technique is contrasted with the other current methods mentioned above. The suggested technique exhibits a superior result for equally distributing the load across nodes, as shown in Fig. 3, which illustrates the acquired values of the makespan for several tasks for the various methods with the suggested technique. The outcomes demonstrate that the suggested method responds faster than alternative algorithms. The simulation findings show that when the number of tasks increases, the effectiveness of other comparison methods on makespan declines. But the suggested approach outperforms the competition well with the value of 120s. The suggested LBMBPSO technique performs superior because it can use resources effectively by distributing loads across the appropriate VMs.

Response Time Comparison Results

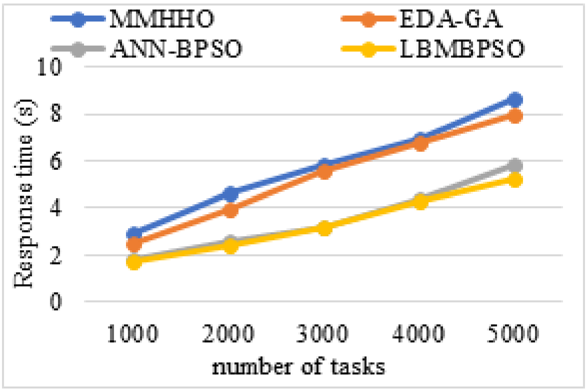

Fig.4. Response time comparison results

The response times of the MMHHO, EDA-GA, ANN-BPSO and LBMBPSO techniques in various tasks are shown in Fig. 4. For the sake of the simulation, tasks are considered autonomous and non-preemptive. The suggested approach is used to dynamically schedule separate jobs. The size of the tasks has an impact on the response time. According to the data, LBMBPSO has a substantially faster reaction time than MMHHO, EDA-GA and ANN-BPSO, which consume energy at rates of 8.63 seconds, 7.96 seconds, and 5.81 seconds respectively. Additionally, it is shown that MMHHO, EDA-

GA and ANN-BPSO all have longer reaction times. Based on the LBMBPSO's outstanding load balancing and quicker reaction times with value of 5.21seconds. However, as the quantity of tasks increases, LBMBPSO is shown to be more efficient than MMHHO, EDA-GA and ANN-BPSO. The average percentage of reaction time is maximum during the tasks due to a rather slightly increased PDR and the truth is that response time employed in the early phases of LBMBPSO with a unique data transmission stage is regarded as overhead energy. The suggested methodology makes use of a suitable load-balancing method to distribute tasks onto virtual machines.

Resource utilization Comparison Results

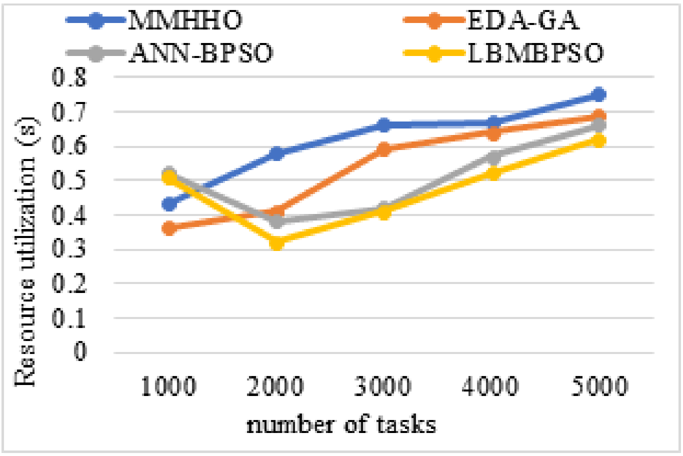

Fig.5. Resource utilization comparison results

The resource requirements of the above-mentioned algorithms are compared to those of the suggested technique. The suggested technique exhibits a greater benefit for distributing the load across nodes equally, as shown in Fig. 5, which depicts the acquired estimates of the resource usage for a quantity of cloud data centers with value of 0.62s for the various methods with the suggested technique. The outcomes demonstrate that, in comparison to other algorithms, the suggested technique has superior resource consumption. The simulation findings show that when the number of tasks grows, other comparison methods perform better in terms of resource consumption. But the suggested approach outperforms the competition well. Because it can efficiently utilize the resources by distributing the loads onto the appropriate VMs using the LBMBPSO approach, the suggested algorithm performs better. Due to the decrease in resource use and the heuristic information employed in the algorithm, the productivity of resources is considerably exploited in the condition of resource utilization.

Average waiting time Comparison Results

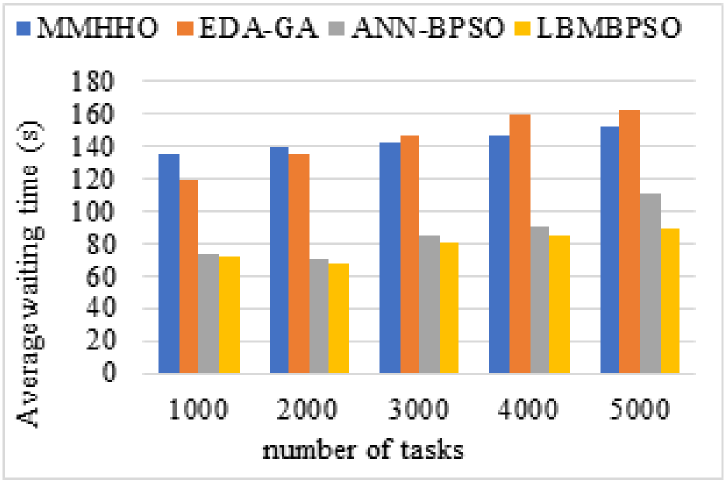

Fig.6. Average waiting time Comparison Results

Comparisons are made between the suggested approach and the aforementioned existing methods in terms of runtime. Thus, according to Fig.6, the suggested technique demonstrates a superior result for uniformly distributing the load over all of the nodes. This figure indicates the acquired estimates of the completion time for a handful of virtual machines for the various methods using the proposed algorithm. The outcomes demonstrate that the suggested method executes more quickly than alternative methods. The simulation findings show that as the number of tasks increases, other comparison algorithms perform better in terms of average waiting time. But the suggested approach outperforms the competition well with value of 89s. It demonstrates how well the suggested method distributes the loads among the VMs and how much the degree of instability is diminished.

Total Cost Comparisons Results

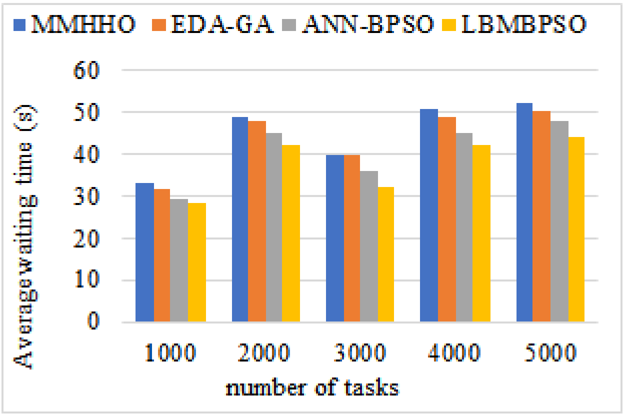

Fig.7. Total cost Comparison Results

The total cost of the MMHHO, EDA-GA, ANN-BPSO and LBMBPSO techniques in various tasks are shown in Fig. 7. There will be less waiting time for each job to be assigned to a virtual machine, which will result in faster processing times. For 5000 tasks, the ANN-BPSO, LBMBPSO method has a total cost of 48 and 44 respectively, whereas the MMHHO, EDA-GA, methods have total cost of 52, and 50.14, respectively. From the results the proposed LBMBPSO is effective and suitable for the cloud scheduling process.

Degree of Imbalance Performance Comparisons

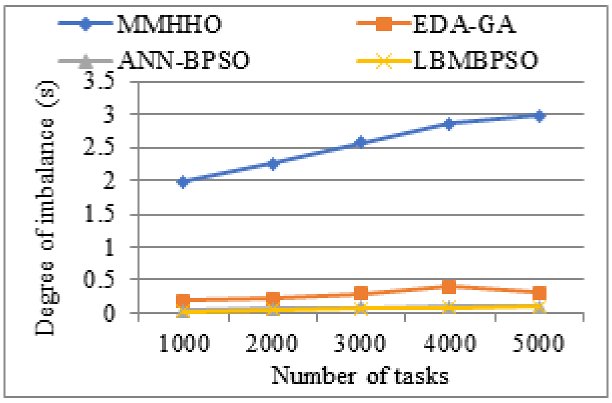

Fig.8. Energy consumption comparison results

As demonstrated in Fig.8, the LBMBPSO method has a lower degree of imbalance than the MMHHO, EDA-GA, ANN-BPSO methods. As a result, the proposed approach outperforms the other current methods in terms of load balancing. For 5000 tasks, the DI for the LBMBPSO method is 0.0896 s while it is 2.983s, 0.324 s, 0.0998s, for MMHHO, EDA-GA, ANN-BPSO, respectively. This indicates that the presented modified BPSO with DNA concept method generates better quality solutions, because the proposed method reducing the DI where it finds an optimum solution and thus significantly reducing energy.

CONCLUSION AND FUTURE WORK

Cloud computing is believed to have great potential in satisfying diverse computing demand from both individuals and enterprises. Considering the virtualized environment in cloud data centers, in this work proposed a load balancing task scheduling algorithm for cloud computing environments based on the binary hybrid gravitational search and particle swarm optimization strategy. It balances the load of application requests submitted from cloud users over virtual machines in the cloud. The proposed algorithm enhances the overall VM utilization of the cloud system. This unique technique is faster than heuristic algorithms in real-world computing environments with low temporal complexity, which benefits consumers by reducing request wait times. The proposed method successfully balances the load, schedules work, and makes the system scalable. This work compared the proposed hybrid algorithm with the pure LBMBPSO. Results show that as the load increases over time, the processing speed of submitted applications decreases, which proves that the proposed LBMBPSO is more efficient in keeping the load balanced over time Moreover, when the number of tasks increases the resource utilization gets increased for all cases and working hard to develop a new technique which can improve the QoS parameters in the near future. As another future work, the proposed scheduler will be applied to scheduling workflow applications with multiple optimization objectives in the cloud environment, including load balancing and energy consumption. And we will do some experiments and compare them with some other algorithms. Also, we hope to apply the proposed approaches to more systems, such as Fog computing.

REFERENCES

1. Choo, K. K. R. (2010). Cloud computing: Challenges and future directions. Trends and Issues in Crime and Criminal justice, (400), 1-6.
2. Xing, Y., & Zhan, Y. (2012). Virtualization and cloud computing. In Future wireless networks and information systems (pp. 305-312). Springer, Berlin, Heidelberg.
3. Masdari, M., Nabavi, S. S., & Ahmadi, V. (2016). An overview of virtual machine placement schemes in cloud computing. Journal of Network and Computer Applications, 66, 106-127.

4. Abid, A., Manzoor, M. F., Farooq, M. S., Farooq, U., & Hussain, M. (2020). Challenges and issues of resource allocation techniques in cloud computing. KSII Transactions on Internet and Information Systems (TIIS), 14(7), 2815-2839.

5. Rashid, A., & Chaturvedi, A. (2019). Cloud computing characteristics and services: a brief review. International Journal of Computer Sciences and Engineering, 7(2), 421-426.

6. Dash, S. B., Saini, H., Panda, T. C., & Mishra, A. (2014). Service level agreement assurance in cloud computing: a trust issue. International Journal of Computer Science and Information Technologies, 5(3), 2899-2906.

7. Li, H., Zhu, G., Cui, C., Tang, H., Dou, Y., & He, C. (2016). Energy-efficient migration and consolidation algorithm of virtual machines in data centers for cloud computing. Computing, 98(3), 303-317.

8. Mathew, T., Sekaran, K. C., & Jose, J. (2014, September). Study and analysis of various task scheduling algorithms in the cloud computing environment. In 2014 International conference on advances in computing, communications and informatics (ICACCI) (pp. 658-664). IEEE.

9. Randles, M., Lamb, D., & Taleb-Bendiab, A. (2010, April). A comparative study into distributed load balancing algorithms for cloud computing. In 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (pp. 551-556). IEEE.

10. Magalhães, D., Calheiros, R. N., Buyya, R., & Gomes, D. G. (2015). Workload modeling for resource usage analysis and simulation in cloud computing. Computers & Electrical Engineering, 47, 69-81.

11. Kansal, N. J., & Chana, I. (2012). Existing load balancing techniques in cloud computing: a systematic review. Journal of Information Systems and Communication, 3(1), 87.

12. Cho, K. M., Tsai, P. W., Tsai, C. W., & Yang, C. S. (2015). A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing. Neural Computing and Applications, 26(6), 1297-1309.

13. Pradhan, A., Bisoy, S. K., & Das, A. (2021). A survey on PSO based meta-heuristic scheduling mechanism in cloud computing environment. Journal of King Saud University-Computer and Information Sciences.

14. Haris, M., & Zubair, S. (2022). Mantaray modified multi-objective Harris hawk optimization algorithm expedites optimal load balancing in cloud computing. Journal of King Saud University-Computer and Information Sciences, 34(10), 9696-9709.

15. Narwal, A., & Dhingra, S. (2022). A novel approach for Credit-Based Resource Aware Load Balancing algorithm (CB-RALB-SA) for scheduling jobs in cloud computing. Data & Knowledge Engineering, 102138.

16. Zade, B. M. H., & Mansouri, N. (2022). Improved red fox optimizer with fuzzy theory and game theory for task scheduling in cloud environment. Journal of Computational Science, 63, 101805.

17. Manikandan, N., Gobalakrishnan, N., & Pradeep, K. (2022). Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment. Computer Communications, 187, 35-44.

18. Mapetu, J. P. B., Chen, Z., & Kong, L. (2019). Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing. Applied Intelligence, 49(9), 3308-3330.

19. Venkataraman, N. (2019). Threshold based multi-objective memetic optimized round robin scheduling for resource efficient load balancing in cloud. Mobile Networks and Applications, 24(4), 1214-1225.

20. Prassanna, J., & Venkataraman, N. (2021). Adaptive regressive holt–winters workload prediction and firefly optimized lottery scheduling for load balancing in cloud. Wireless Networks, 27(8), 5597-5615.

21. Nabi, S., Ibrahim, M., & Jimenez, J. M. (2021). DRALBA: dynamic and resource aware load balanced scheduling approach for cloud computing. IEEE Access, 9, 61283-61297.

22. Kruekaew, B., & Kimpan, W. (2022). Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm With Reinforcement Learning. IEEE Access, 10, 17803-17818.

23. Ali, I. M., Sallam, K. M., Moustafa, N., Chakraborty, R., Ryan, M., & Choo, K. K. R. (2020). An automated task scheduling model using non-dominated sorting genetic algorithm ii for fog-cloud systems. IEEE Transactions on Cloud Computing, 10(4), 2294-2308.

24. Pang, S., Li, W., He, H., Shan, Z., & Wang, X. (2019). An EDA-GA hybrid algorithm for multi-objective task scheduling in cloud computing. IEEE Access, 7, 146379-146389.

25. Alghamdi, M. I. (2022). Optimization of Load Balancing and Task Scheduling in Cloud Computing Environments Using Artificial Neural Networks-Based Binary Particle Swarm Optimization (BPSO). Sustainability, 14(19), 11982.