The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 11 Issue 1, 2023
pp: 2111 - 2124

A Novel Heterogenous Dominant Sequence Clustering for Task Scheduling and Optimal Load Balancing Using JSO in Cloud

B. Kalaiselvi
Research Scholar, Mother Teresa Women's University, Kodaikanal
Dr. M. Pushparani
Professor and Head, Department of Computer Science,
Mother Teresa Women's University, Madurai

Abstract: Cloud computing (CC) is rapidly increasing and being used more and more in information technology (IT) contexts. Among the hottest issues in the world of CC is task scheduling. In cloud datacentres, load balancing is accomplished using a variety of scheduling techniques, although lengthening the parallel time. By taking task duration and VM capacity into account, this study proposes a cluster-based task scheduling paradigm. The makespan and execution duration should be reduced, the suggested system engages in dynamic load balancing. In this study, we suggest a unique method called HDDSC-JSO, which employs the Jellyfish Swarm Optimization (JSO) algorithm for load balancing and heterogeneous Density based dominant sequence clustering (HDDSC) for job scheduling. Using the HDSC technique, It presents a graph of one or more groups representing user tasks, the tasks of users are first clustered. The Adapted Diverse Expeditious Termination Time (ADETT) method is used to score each work after task clustering. whenever the job with the greatest priority is scheduled first. Next, using a JSO algorithm, load balancing is carried out, distributing the load according to the weight and capacity of the server as well as the connection of the client to the server. Task allocation chooses a heavily weighted or unconnected server, lengthening the response time. Finally, measures including response time, makespan, resource consumption, and service dependability were used to assess the suggested architecture.
Keywords: Cloud Computing, Load Balancing, Task Scheduling, Heterogenous Density based Dominant Sequence Clustering, Jellyfish Swarm Optimization and VMs.

INTRODUCTION

In recent years, network technology has undergone a critical revolution thanks to cloud computing. The fundamental concept behind cloud computing is to offer customers with a variety of associated services via the collaborative usage of shared hardware resources and software applications. There are three different forms of cloud computing architecture, with software as a service (SaaS) being the first [1]. The system may supply software resources on the platform in SaaS architecture to make it easier for consumers to utilize. Platform as a Service (PaaS) is the second kind, and it is the one that allows users to build and write their applications via the platform [3]. Infrastructure as a Service (IaaS) is the last type [4]. Users don't need to buy hardware since it enables them to store, carry out, and compute tasks. Storage, memory, computation, and bandwidth are all provided by cloud data centres [5] in order to handle user requests and duties. Using a promising technique called virtualization, it furthermore provides simple, a shared pool of flexible computing resources that is accessible across the network on demand. The fundamental element of cloud computing is virtualization. It is a well-known technology that provides processing units in the form of virtual resources known as VMs to allow cloud computing [6]. Resources may be scaled up or down in the cloud computing environment by adding or deleting virtual machine instances. In the cloud computing environment, where VMs operate concurrently, these VMs carry out the individual tasks supplied by the customers [7]. Thus, these tasks must be scheduled in a manner that maximizes the use of available resources.

Cloud computing strategies provide consumers numerous benefits, yet there are also many difficulties and issues. In a cloud computing environment where efficiency is vital, task scheduling is one of the most popular study areas. In cloud computing settings, several academics have so far suggested a wide variety of job scheduling techniques. Both dynamic and static scheduling techniques may be categorized using these algorithms. Using the dynamic scheduling approach, which is ideal for real-time task assignments, the system will assign jobs without knowing when they will be completed. Ant Colony Optimization (ACO) is one of the standard dynamic scheduling techniques [8], Genetic Algorithm (GA) [9], Particle Swarm Optimization (PSO) [10], and Simulated Annealing (SA) [11]. Static scheduling techniques are superior than dynamic scheduling techniques [12] Use known information in advance, such as min-min, to dispatch the jobs with minimum overhead [13], techniques

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 11 Issue 1, 2023
pp: 2111 - 2124

such as max-min [14] and Suffrage. Hence, compared to dynamic scheduling systems, static ones may achieve superior performance and load balancing.

According to the findings of studies in the literature [15], most cloud computing environments now utilize only basic scheduling algorithms, which might result in unequal load distribution during job scheduling and resource allocation. These scheduling algorithms either pick dispatching nodes with the lowest value of trust and benefit, or they choose dispatching nodes with the highest value of trust and benefit. They also either dispatch jobs with the earliest completion times to the relevant resources. The connection between node load and task distribution is not handled by them. The system's overall performance will suffer if certain unbalanced or overloaded nodes are chosen. Taking into consideration load balancing, task execution time and cost, and the discussion from above, this work tries to complete a task scheduling. Certain load balancing scheduling techniques fail to take into account the fact that a system's overall performance may suffer if activities carried out by underperforming nodes, particularly when there are a lot of diverse tasks. In order to reduce the makespan, execution time, and workload variance across the VMs, a novel scheduling approach called cluster-based task scheduling is presented. These are the main contributions of this study, which mainly tries to increase response time via work scheduling utilizing clustering and optimal load balancing algorithms:

- To schedule user duties, the HDDSC algorithm is used, which displays upcoming jobs as graphs. Each graph has more than one cluster. Also, the ADETT algorithm is utilized to order planned tasks, scheduling the job with the greatest priority first for the next procedure.
- Tasks are assigned to VMs that sequentially reduce response time using the JSO algorithm, which takes server capacity and client connection into account.

The remaining sections are arranged as follows: The summary of relevant work in Section 2. Section 3 provides an explanation of the recommended methodology employed in this research. Details regarding the experiments and discussions are given in Section 4. The paper's conclusion and recommendations for further study are provided in Section 5.

RELATED WORK

Mangalampalli et al., [16] created a multi-objective, trust-aware scheduler that prioritizes jobs and virtual machines and assigns them to the best available virtual resources while reducing wait times and energy usage. MOTSWAO is the name of the whale optimization algorithm that we utilized to simulate our task scheduler. While creating a scheduler, this study carefully considered makespan, which is a crucial viewpoint for any cloud paradigm, but energy consumption reduction is also a crucial parameter that benefits both cloud users and cloud service providers. A deep reinforcement learning model based on incentives and feedback is needed to schedule workload onto virtual resources and optimize parameters.

Guo [17] a simulation experiment tests the efficiency of the fuzzy self-defense algorithm-designed cloud computing multi-objective work scheduling optimization algorithm. With the lowest possible scheduling completion time, it thereby achieves a superior scheduling impact. As the number of cloud computing multi-objective tasks increases, the suggested approach maintains a low deadline violation rate, around 5% lower than previous methods. Nevertheless, no complete solution has been developed that adequately satisfies these criteria.

Talha et al., [18] using the Pathfinder algorithm (PFA) and the oppositional-based learning algorithm (OBL), a new cloud workflow scheduling method. Three objectives were prioritized by the suggested scheduler. The workflows toolkit is used for simulation testing to gauge the viability of the proposed strategy. The approach of scheduling that has been provided is designed to ensure optimum resource utilization across VMs while needing the least amount of expense and time as possible. While there are more multi-objective tasks assigned to the VM, the load balancing impacts may not be improved for the cloud computing optimized work on the VM.

Sharma & Garg [19] built an event-based ACO model to manage dynamic data allocation in order to fit the resource limits and activity schedule. EBS with ACO is not a suitable method for multi-objective scheduling. The proposed multi-objective strategy can organize an activity with enough tasks and resources. The test results show that LBACO optimizes the objective function fastest and most efficiently. Compared to other methods, ACO can create better plans with better statistics, mean access times, and task assignments. The personnel allocation matrix, however, is not taken into consideration when scheduling projects using the conventional method.

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 11 Issue 1, 2023
pp: 2111 - 2124

Mahmoud et al., [20] the HEFT algorithm was modified to accommodate load balancing. Load Balancing HEFT (LB-HEFT) is the name of the updated algorithm. LB-HEFT, E-HEFT, and HEFT algorithms were compared to evaluate their efficacy. The novel LB-HEFT algorithm outperforms the existing E-HEFT and HEFT algorithms by improving load balancing by 43.49% and 72.59%, respectively, resource usage by 2.28% and 5.61%, and makespan by 7.55% and 3.75%. Dynamic load balancing increases inter-VM communication overheads, a major issue (swapping files between VMs). Load balancing approaches overlook communication overheads.

Nabi et al., [21] proposed using an approach for dynamic task scheduling that takes into account the available resources. There are three well-known datasets in this example, namely HCSP, GoCJ, and Synthetic workload, were taken into consideration for the simulation tests using the Cloudsim simulation tool. Since the user must pay for a resource, the task scheduling issue is regarded as one of the major obstacles. Finding the ideal schedule is difficult since the task scheduling issue is NP-hard in nature.

Kruekaew & Kimpan [22] presented the MOABCQ method, It combines the Artificial Bee Colony Algorithm (ABC) with the reinforcement learning technique known as Q-learning to speed up the performance of the ABC algorithm. It is an independent job scheduling solution for cloud computing. The recommended strategy aims to increase VM throughput, balance load across VMs based on makespan, cost, and resource use, and remove the limitations of concurrent concerns. The presence of a problem that might impede processing or result in a system crash during cloud access is a host or server that is overcrowded or underloaded.

Ali et al., [23] to handle the discrete multi-objective task-scheduling problem and automatically assign tasks that should be completed on fog or cloud nodes, a discrete non-dominated sorting genetic algorithm II (DNSGA-II)-based optimization model was created. Instead of employing continuous operators that take a lot of computing power and are unable to assign enough computer nodes, the NSGA-II method has been modified to discretize the evolutionary operators for crossover and mutation. The suggested paradigm distributes processing power that may be used to successfully operate on either fog nodes or cloud nodes. Moreover, it effectively arranges task allocation among diverse computer resources in the fog. Work should be distributed across all VMs in parallel to balance the system and make the most use of the resources available.

Pang et al., [24] based on EDA (estimation of distribution algorithm) and (genetic algorithm) GA, creates an EDA-GA hybrid scheduling algorithm. To start, a predetermined range of feasible options is produced using the probability model and sampling technique of EDA. Second, the search space of solutions is expanded using the crossover and mutation GA procedures. The ideal task scheduling method for VMs is at last implemented. Unfortunately, there were some issues with the distribution of the load and the use of the available resources.

Zhao et al., [25] Load Balancing based on Bayes and Clustering (LB-BC), a unique heuristic technique, was presented to address the challenge of choosing physical hosts for deploying desired workloads. The majority of earlier studies typically make use of a number of algorithms by selecting the best suitable target hosts inside an algorithm cycle and immediately implementing load balancing on those hosts. In order to eventually acquire the ideal clustering set of physical hosts, the Bayes theorem is used with the clustering process.

Summary: As was previously said, it was shown to be of utmost importance to design efficient algorithms for job scheduling and resource selection in cloud computing settings. Studies that take into account single goal, bi-objective, and multi-objective scheduling put a lot of work into task organization or resource allocation for each activity. Since both makespan and processing time and cost goals match the expectations of the users, earlier research mostly focused on these objectives, but this study introduced multi-objective optimization of the task scheduling problem. To operate with cloud computing, however, a number of factors or objectives must be taken into account. By adopting the resource cluster-based load method, which uses a clustering algorithm to reduce the needed time and cost and minimize needless moves by dynamically allocating different weights to different resources depending on their utilization intensity.

PROPOSED METHODOLOGY

In this work, mainly applied two algorithms: HDDSC algorithm for task scheduling and JSO algorithm for load balancing that mainly aims to do the following: Techniques for task scheduling and load balancing are used so that the load may be more evenly distributed among virtual nodes and the response time (RT) can be reduced. The planned work is shown in Figure, where jobs were scheduled using the HDDSC algorithm in accordance with

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 11 Issue 1, 2023
pp: 2111 - 2124

their priorities. The deadline and makespan determine a task's priority. Each task is rescheduled and given a priority rating for the following steps in order to identify which is the greatest priority. After that, VMs are clustered. According to the flowchart for our proposed work, the HDDSC algorithm receives user tasks directly and ranks them according to priority using the ADETT algorithm. The VM allocation mechanism is used to move the job with the greatest priority. In order to balance loads, the JSO algorithm takes into account the capacity and client connections of the servers. Tasks are then distributed among VMs in a way that gradually reduces response time.

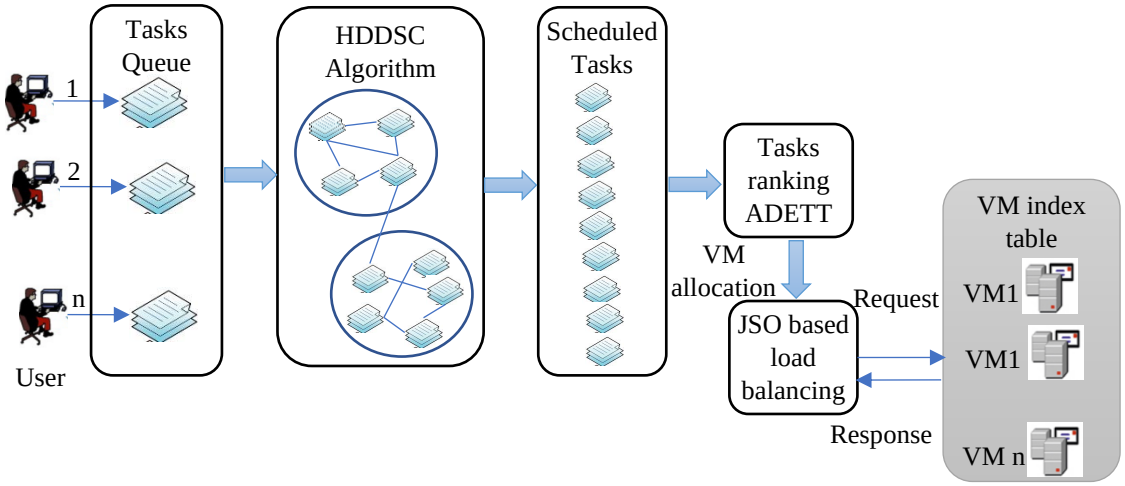

Fig.1. The Architecture of the proposed Task Scheduling and Optimal Load Balancing

Preliminaries of the System Model for Task Scheduling and Load Balancing

For more clarity regarding cloud computing's issue with task scheduling, let us consider a cloud computing system that , which is represented as m heterogeneous independent VMs, $VM = \{VM_1, VM_2, \ldots VM_n\}$, and a set of n tasks that are represented by $T = \{T_1, T_2, \ldots T_n\}$.

Resource Model: A resource model is applied in order to describe the computing power of individual nodes. A single resource model is built so that the performance of each node may be represented and used as a reference for the scheduling of tasks. The capacity of the resource measured in a time unit may be used to talk about its capabilities in the areas of compute, storage, and network transmission.

Task Model: When a task is carried out by a node, the resources are really used by the task. According to the node's resource index, calculated is the task's resource consumption index. To determine the number of tasks, The resource vectors' weight sum is determined, It serves as the task's t_i denotation i.

$$T_i = RV_i \times W^T \quad (1)$$

RV_i is the task that uses the resource vector, and W is referred to as the vector of resource weights. Assume that task i which is being run in VM j, uses up all of VM j resources. In this case, the time required to finish job i equals the task's execution time in VM j.

$$ET_i^j = \frac{T_i}{RV_i} \quad (2)$$

Load Model Definition: The load of node i at time t is indicated as the following to show the load state of a node: LD_i^t. The VM status table is set as set VM Each VM_i in VM is a status tuple of the VM j and I is the length of the task before execution.

$$LD_i^t = \frac{VM_i \cdot I}{\Delta t} \times W^T \quad (3)$$

The cluster load is written as, and it is used to determine on the average load of each node, the loads of the cluster LD^t at the time t.

$$LD^t = \frac{\sum LD_i^t}{n}, i \in cluster\ numbers \quad (4)$$

The prediction load PLD^t The load prediction technique allows for the acquisition of at the moment t.

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 11 Issue 1, 2023
pp: 2111 - 2124

HDDSC Method for task scheduling

The cluster load LD^t at the time t using the virtual status database, it may be calculated. the current cluster load $t+1$ is computed with PLD^t and LD^t the load prediction method is used. Here, the HDDSC technique is employed for clustering. Algorithm 1 and Fig. 2 both provide a flowchart and an algorithmic explanation of the HDDSC algorithm. The design of the HDDSC algorithm uses a Heterogeneous Enumerate Scheduling (HES) method. Using priority computation, the DSC method groups upcoming tasks. P_i placing the greatest value top_i tasks' top, middle, and bottom tiers. From the entry-level work to the highest level of a job, the length of the longest path is defined by i. The lowest level $bottom_i$ the separation along the longest path from i to exit task. The following formula is used to calculate task priority:

$$P_i = top_i + bottom_i \quad (5)$$

To determine a task's priority, the top and lowest levels of the job are put together P_i. The priority values for each of the tasks are generated and then used in the task clustering process, which is then followed by the task scheduling process. Using the DSC method, tasks are grouped according to their priority. To determine the ideal priority work to be shifted to the next processes, scheduled tasks are graded. The density $\mathfrak{D}(i)$ is established as follows: The distances between each task i and the other tasks are computed, and they are sorted in increasing order. The jth VM standard deviation Σ_i^j, j is often selected as the square root of the total size of the VM, and it is utilized for $\mathfrak{D}(i)$ calculation:

$$\mathfrak{D}(i) = \frac{1}{\Sigma_i^j + 2} \times W^T \quad (6)$$

To be scheduled in phase two of the HES phase is an unscheduled free task number. As a priority of priority list construction, the algorithm chooses the task by combining the top level and bottom level of tasks. a task's highest level n_i $(toplevel(n_i))$ is characterized as the distance along the longest route from a top-level entry task to n_j and the last stage of an assignment n_j $(bottomlevel(n_i))$ is described as the distance along the longest route from ni to a bottom job. It is challenging to identify the $toplevel(n_i)$ and $bottomlevel(n_i)$ for activities in a heterogeneous environment that have not been scheduled into processors. The HES technique's proposed answer is to create a unit cluster with the following formulation of the goal function for those jobs that have not yet been scheduled:

$$OF(i) = \min \Sigma(\mathrm{i}) \quad (7)$$

where Σ is the standard deviation in workload and it calculated based on LD^t, ET_i^j, P_i, $\mathfrak{D}(i)$ and task length T_i as follows:

$$\Sigma = \sqrt{\frac{1}{N}\sum_{j=1}^{n}(LD^t - \overline{LD}^t)} \times ET_i^j \times T_i \times P_i \times \mathfrak{D}(i) \quad (8)$$

The CPU resource is prioritized above other resources since the tasks need a lot of computation. As a result, priority values may be established even when tasks in each scheduling phase have not already been scheduled. This allows for the creation of a prioritized list of open tasks. To allocate a job to a processor, Choose from the list the free task n with the greatest priority. The task n_x, based on rating by a reduction technique and scheduled into a processor. The minimization process seeks to reduce task completion times so that scheduling steps may be completed with the least amount of parallel time possible. Step 1 of the reduction process is used to identify a processor in order to reduce the completion time pr_i Set PS on the CPU to enable the best possible job completion. Parallel time will then be further reduced by lowering the number of $toplevel(n_x)$ in the reduction process's phases 2 and 3. The plan to cut down even more $to\ plevel(n_x)$ is to schedule the previous tasks so as to minimize the communication edges between the job n and those tasks n_x, into n_x's cluster where this measure would not result in an increase $toplevel(n_x)$. When scheduling a predecessor, this activity might cause a scheduling conflict n whose children are different from n_x. This increases the complexity of time and the algorithm. Schedule only predecessors of these to avoid them $n_x$in which n_x, their only child.

To order scheduled tasks produced by the DSC algorithm, the ADETT algorithm is employed. For impending procedures like VM clustering and load balancing, the planned job with the greatest priority is chosen. Average execution and transmission times for data transfers are calculated to rank tasks. ADETT, a static list algorithm that prioritizes tasks that significantly affect process execution time, is the best task prioritization method.

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 11 Issue 1, 2023
pp: 2111 - 2124

Procedures for mapping issues employ static methods that accept static circumstances for a given duration. The critical route of the work calculates the upward rank of each task based on its maximum communication time and average implementation time from start to finish. If two tasks have the same priority and arrival time is confirmed, the first task gets precedence. The rank of task $R(T_i)$ is illustrated by the following examples of how communication time is used:

$$R(T_i) = ET_i^j + \max_{T_i}\left(al + \frac{data_i^j}{bandwidth} + al + \Sigma \cdot R(T_j)\right) \quad (9)$$

Here $CT_i^j = al + \frac{data_i^j}{bandwidth}$ the period of time during which messages are sent in order to transmit $data_i^j$ via edges i and j, where al where is the average delay $bandwidth$ is the system's average communication connection bandwidth across VMs, ET_i^j is the task's average time to complete across all VMs, and $R(T_j)$ is calculated using all of its offspring.

Algorithm 1. The HDDSC step-by-step job scheduling process

Sort the CPUs that are available using *PS*, then sort them according to speed.
The collection of tasks in the scheduling task graph is called *T* in the examined graph.
The assumption is that each task constitutes a single processor cluster with load and task length, and each task is tagged as unreviewed.
Calculate each task's top level and bottom level, then for each free task, set the top level to zero.
Create a priority list of open tasks with the tasks ordered in decreasing order of priority values using the sum of a task's top level and bottom level as the priority value.
Decide which task should come first n, from T_i.
Use the ranking-based minimizing approach to lower the highest level of n_x.
Between two separate activities that are placed into the same cluster depending on density, add pseudo edges.
Mark the task examined and remove n and put n_x.
Recompute the priorities of the n_x's successors tasks.
Schedule tasks in the set T_i into P_i such that $toplevel(n_x)$ is minimum when n_x, is scheduled into P_i.
Schedule n_x, into P_i with the ranking $R(T_i)$ as in Eq.(9).
End while.

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 11 Issue 1, 2023
pp: 2111 - 2124

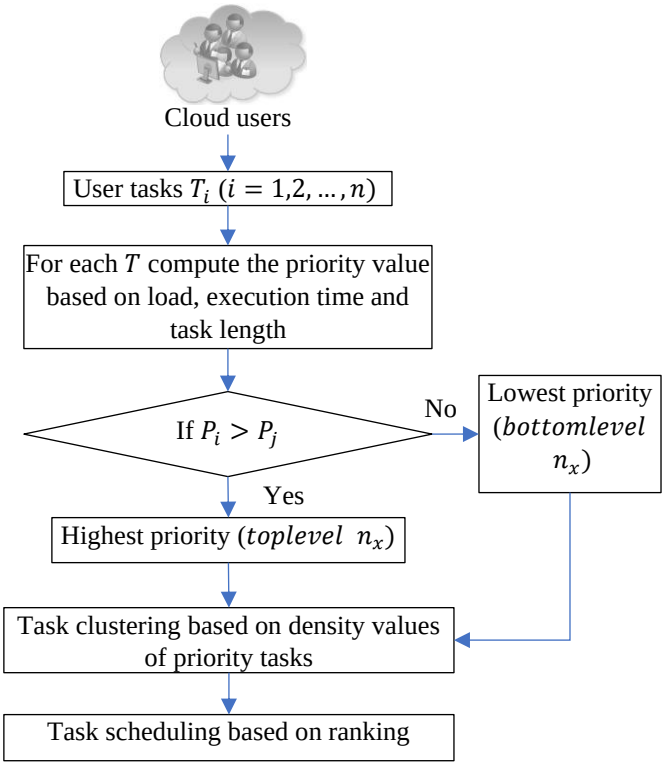

Fig.2.Flowchart of HDDSC based task scheduling

Load balancing Using VMs Optimization

This paper suggests the JSO algorithm, which combines the neighbohood exchange strategy with the jellyfish technique, to lower the cost of energy use. Chou and Truong [26] suggested a jellyfish algorithm that replicates the way swarms of jellyfish act while looking for food. VM optimization is subject to three rules. The first rule states that VM travel in swarms by adhering to ocean currents, and this modification in movement mode is controlled by a "temporal control mechanism". The second rule is that VM will migrate in the water in search of food. People gravitate toward locations with more food. The location and purpose of the destination affect the quantity of food found, according to the third rule. Initializing the location of the VMs and ocean currents are only two of the five main phases of JSO that this research suggests; depending on VM movement, update position; time management system; using the big rank value rules; based on neighbourhood exchange protocols, as well as enhanced solutions. The next part provides further information on the proposed SSO algorithm process.

Every overloaded VM should have access to the resources it needs since the system should be aware of this demand. According to capacity, the VMs are arranged in increasing order. The suggested technique chooses the job from the overcrowded VM with the lowest priority and reschedules it to an underloaded VM (target VM) with the best capacity. Equation may be used to compute the supply to a particular VM, which is the difference between its capacity and the present load (3 and 4). In such case, a VM's demand is determined as $LD^t - capacity_{VM}$ the system determines the standard deviation when each work is uploaded to the cloud. The system is balanced if the VM load's standard deviation falls below or meets the threshold condition, which is defined as having a value between [0-1]. If not, the system is out of balance, overloaded, or underloaded. The load balancing procedure is started if the standard deviation of the loads exceeds the threshold. VMs are divided into underloaded and fully laden VM groupings throughout this load balancing process. Afterwards, the duties are redistributed to the VM with the greatest capacity. A VM cluster is overloaded when its current workload exceeds the group's maximum capacity, which is determined by the threshold value. In this VM cluster, load balancing is not possible.

Initialization of the VMs: This section explains how the location of the virtual machines is determined using the logistics map, where, X_i is the i−th A chaotic logistic value exists at the VM location. A method was used to create the VM's initial population X_0 it does not have components but parameters (0, 1) (0, 0.25, 0.5, 0.75, 1). ϵ has a value of 4 for the parameter value.

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 11 Issue 1, 2023
pp: 2111 - 2124

$X_{i+1} = \epsilon * X_i(1 - X_i), 0 \leq X_0 \leq 1$ (10)

The formulation of sea currents is found in (16), $rand$ is random number, ρ is the probability of the distribution exceeding 0 and also X_{cur} is the ideal VM site right now. The location of every VM on average is shown by AL.

$X_i(t+1) = (X_i(t) + rand(0,1) \times X_{cur}) - (\rho \times rand(0,1) \times \mathrm{AL}) + X_i(t)$ (11)

Position updating based on VM movement: Position updates depending on the movement of the VM are described in this section. Active and passive motions dictate how the VMs move inside the swarm. For instance, passive motion means the VM just stands still. The result of the passive motion shows the new position of VM (17). In situations when a search space is constrained by an upper constraint, $\mathfrak{Ub}$ the lower limit by, and $\mathfrak{Lb}$. A coefficient of motion is used to represent the length of motion around the VM point $\mathfrak{c}$ larger than zero in value. Active mobility, on the other hand, is seen as successful local search space utilization. In order to define it, use the formula indicated in (18).

$X_i(t+1) = X_i(t) + \mathfrak{c} \times rand(0,1) \times (\mathfrak{Ub} - \mathfrak{Lb})$ (12)

$X_i(t+1) = X_i(t) + rand(0,1) \times \overrightarrow{\mathfrak{D}(\iota)}$ (13)

$$\overrightarrow{\mathfrak{D}(\iota)} = \begin{cases} X_j(t) - X_i(t) & if\ OF(X_i) \geq OF(X_j) \\ X_i(t) - X_j(t) & if\ OF(X_i) \geq OF(X_j) \end{cases} \quad (14)$$

Where, $\vec{D}$ dependent on density value, does each VM behave in a way that will help it discover food. In (5), this behavior is indicated OF is the fundamental characteristic of place X.

Process of time control mechanism: The movement of VM caused by variations in the temperature of the sea current, passive and active motions, is modelled using the time control method. Modelled by the march in (15). As time passes, VM swarms grow (iteration). Using both active and passive motions, each VM moves within the swarm to find a better place. Exploitation is the term used to describe this conduct $e(t)$ changes after each repetition. The number of swarms in the proposed method, together with $\max iter$ are the two inputs that this method requires. where the number of iterations allowed is max.

$e(t) = \left(1 - \frac{1}{\max iter}\right) \times \rho(2r(0,1) - 1)$ (15)

Using a high rank value, The VM allocation issue, a discrete problem, uses the journey sequence as the decision variable. This research suggests a high rank rate to modify the location of the VM vector in the journey sequence ($\mathbb{L}$) scheduling and allocation issues are examples of simple approaches that may be used for complex problems. To calculate the overall distribution cost, apply the formula indicated in (16). On the basis of, the mathematical formula for the load balancing issue is offered.

$\min \sum_{i=1}^{n} OF(i)$ (16)

Where VM allocation to task i on n number of tasks.

Apply neighbourhood exchange: Neighbourhood exchange is utilized to enhance the efficiency of the VM algorithm throughout each iteration. In this research, the swap and neighbourhood flip exchange rules are proposed. Two randomly selected VM position vectors are reversed using the flip rule. To illustrate the swap rule, two randomly chosen VM position vectors are switched. The neighbourhood swap process must be done 0.25 times for each iteration. In each iteration, the $\mathbb{L}$ VM locations are transformed into a journey sequence using a method. The optimal allocation solution on the current iteration is determined by comparing the prior solution with the new Neighbourhood exchange result. Because of this, security is attained and less energy is used the greater the population and iteration parameters.

Based on the SD value determined using equation, load balancing and task rescheduling decisions are made at this step (8). Only when the capacity of the data center is larger than the present load will the choice be made in order to ensure system stability by reducing the number of migrations. A threshold value is chosen for the purpose of determining the load (value ranges from 0 to 1), and it is compared to the computed SD measure. Only when the computed SD exceeds the threshold value are load balancing and scheduling carried out. Figure 3 in this part shows a flow diagram illustrating the JSO's control mechanism for load balancing. It explains how load balancing between overloaded and underloaded virtual machines works. The fundamental concept is to send the tasks to the VM until the machine becomes overloaded, which is defined as the load on that VM reaching a value that is higher than the threshold. Here, avoid sending jobs to overcrowded machines and instead send the remaining workload

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 11 Issue 1, 2023
pp: 2111 - 2124

to VMs that can be located using the random search approach since they are underloaded. Here, the tasks/jobs that the user requests to the VM are represented by Cloudlets in the flowchart.

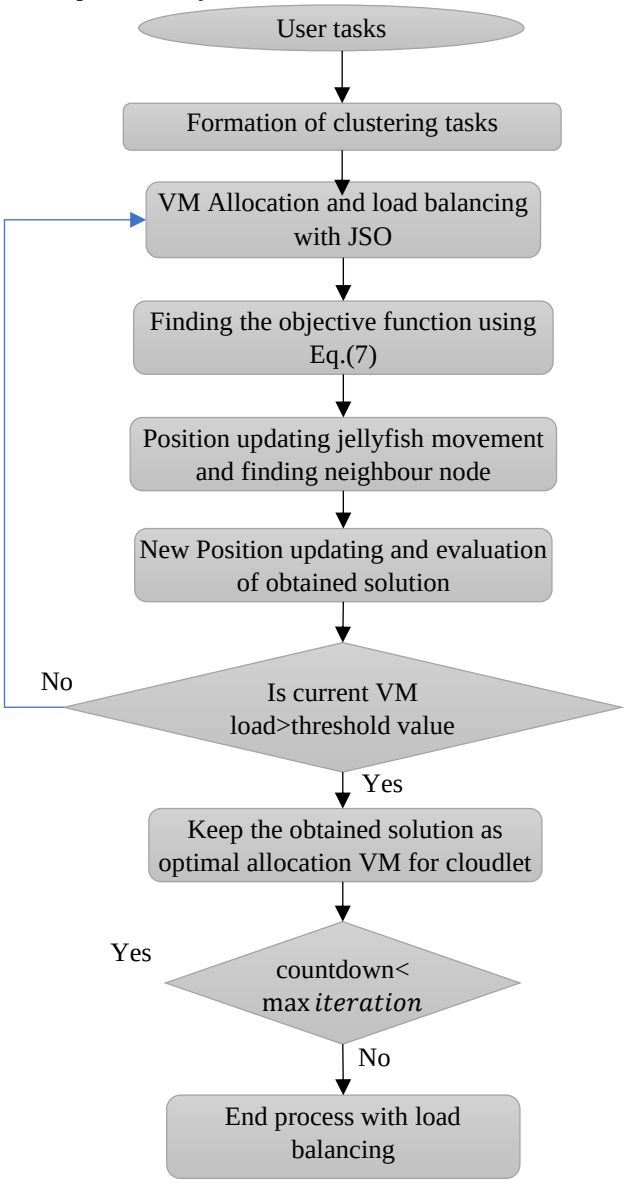

Fig.3. Flowchart of the proposed JSO based load balancing

Experimental Results and discussion

Explain the experimental results and performance assessment of the proposed HDDSC-JSO approach and compare it to existing state-of-the-art scheduling algorithms like MMHHO [27], EDA-GA [28], ANN-BPSO [29], and LBMBPSO. The effectiveness of the proposed method in terms of make-span, typical waiting time, reaction time, degree of imbalance, overall cost, and resource usage was compared to three different scheduling algorithms.

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

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

$$Totalcost = \sum \mathcal{CT}_{ij} \times C_{VM_j} \quad (12)$$

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

Experimental setup: When the infrastructure is inflexible, such in Amazon EC2 or Microsoft Azure, it is hard to test new methods or concepts due to security, speed, and the high dollar cost of repetitive testing. As these tests need a lot of work to make them scalable and repeatable, they are challenging to run on actual cloud

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 11 Issue 1, 2023
pp: 2111 - 2124

infrastructures. According to previous research, the CloudSim-3.0.3 simulator may be used to assess an algorithm's performance in real-world conditions. The simulator itself generated extra data at random and utilised it throughout testing. Using a computer with an Intel Core i5-6500 processor operating at 3.2 GHz and 4GB of Memory, it was feasible to create and run all of the Java code simultaneously. The initial configuration is represented by the first value since all VMs are distributed equally across all hosts. The second result shows the unequal distribution of VMs across hosts.

Makespan comparison results

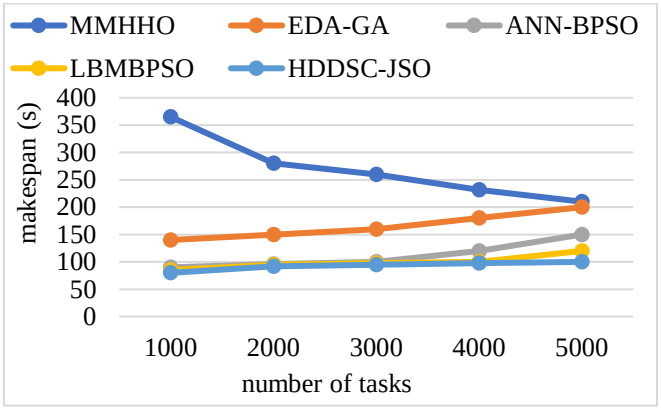

Fig.4. make span comparison results

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. 4, 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 HDDSC-JSO method responds faster than alternative algorithms such as MMHHO, EDA-GA, ANN-BPSO and LBMBPSO. According to the simulation results, as the number of tasks increases, the effectiveness of other comparison methods on makespan declines. But the suggested HDDSC-JSO approach outperforms the competition well with the value of 100s. Since in the cloud, both the quantity of users and the nature of their requests used to be dynamic. So, in order to reduce the time, it takes for requests to be fulfilled, a scheduling technique that takes this heterogeneity into account while allocating requests to relevant resources is required. The suggested scheduling system uses HDDSC, a simple and effective clustering technique, to create task groups based on the similarity in task durations.

Response Time Comparison Results

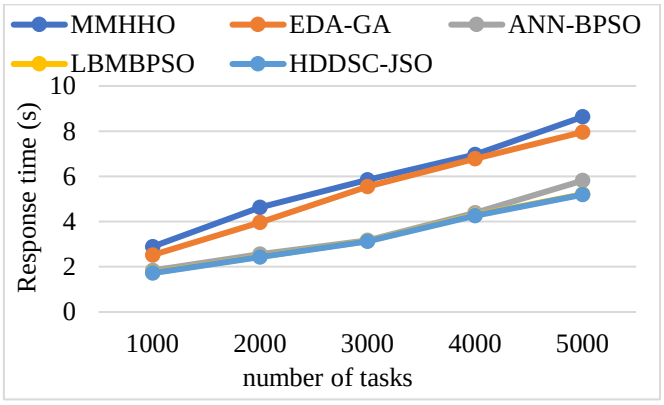

Fig.5. Response time comparison results

The response times of the MMHHO, EDA-GA, ANN-BPSO, LBMBPSO and HDDSC-JSO techniques in various tasks are shown in Fig. 5. For the sake of the simulation, tasks are considered autonomous. The suggested approach is used to dynamically schedule separate jobs. The amount of work involved affects reaction time. According to

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 11 Issue 1, 2023
pp: 2111 - 2124

the data, MMHHO, EDA-GA, ANN-BPSO, LBMBPSO and HDDSC-JSO has a substantially faster reaction time, which consume energy at rates of 8.63 seconds, 7.96 seconds, 5.81 seconds and 5.21 seconds respectively. Additionally, it is shown that MMHHO, EDA-GA, ANN-BPSO and LBMBPSO all have longer reaction times. Based on the HDDSC-JSO's outstanding load balancing and quicker reaction times with value of 5.19 seconds. Reduce the amount of time a task needs to service a VM queue with the help of priority-based balancing. As a result, the method stands up well in terms of a minimum reaction time compared to the standard JSO without adding extra overheads. It also decreases the response time of virtual machines.

Resource utilization Comparison Results

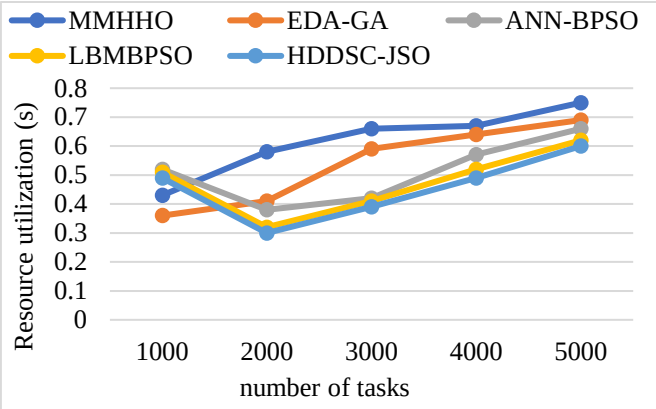

Fig.6. Resource utilization comparison results

Figure 6 illustrates the resource consumption of the MMHHO, EDA-GA, ANN-BPSO, LBMBPSO, and HDDSC-JSO approaches when applied to a variety of different jobs. This will result in speedier processing times as there will be less resource use for each task that is allocated to a VM. For 5000 tasks, the existing methods such as MMHHO, EDA-GA, ANN-BPSO and LBMBPSO, methods have resource utilization of 0.75, 0.69, 0.66 and 0.62, respectively. From the results the proposed HDDSC-JSO is effective and suitable for the cloud scheduling process attained resource utilization of 0.6. Since each scheduling step's parallel time may be reduced by scheduling work into a heterogeneous processor environment using this technique. Also, it is shown in the experiment that when the standard deviation of the processor configurations increases, the algorithm yields a lower scheduling time. By using several scheduling heuristics in a two-phase method, the HES technique may be utilized to construct additional heterogeneous scheduling algorithms in addition to designing HDDSC.

Average waiting time Comparison Results

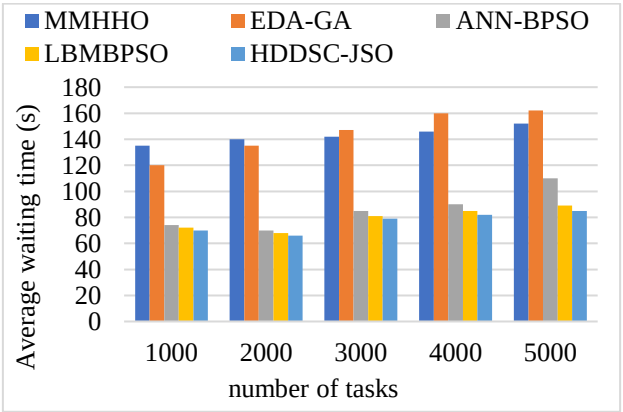

Fig.7. Average waiting time Comparison Results

The average waiting time of the MMHHO, EDA-GA, ANN-BPSO, LBMBPSO and HDDSC-JSO techniques in various tasks are shown in Fig. 7. Faster processing speeds will come from reduced waiting time before each task

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 11 Issue 1, 2023
pp: 2111 - 2124

is allocated to a VM. For 5000 tasks, MMHHO, EDA-GA, ANN-BPSO and LBMBPSO methods have average waiting time of 152s, 162s, 110s and 89s, respectively. From the results the proposed HDDSC-JSO is effective and suitable for the cloud scheduling process with the value of 85s. The suggested architecture reduces VM migration by improving load balancing via JSO allocation while improving average waiting time through task grouping and ranking. As a result, the suggested work's reaction time is shorter and the quality of the service is higher. A plot between task size and trust level is used to display the service dependability of a task. Every task that a user submits to the cloud is assigned a trust value based on the service that will be delivered for their subsequent work.

Total Cost comparisons results

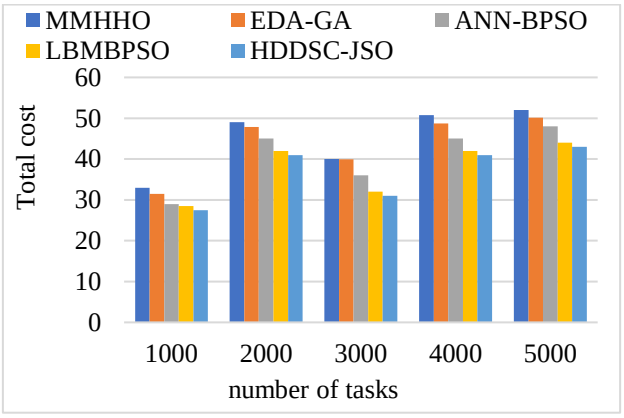

Fig.8. Total cost Comparison Results

The total cost of the MMHHO, EDA-GA, ANN-BPSO, LBMBPSO and HDDSC-JSO techniques in various tasks are shown in Fig. 8. Faster processing speeds will come from reduced waiting time before each task is allocated to a VM. For 5000 tasks, the MMHHO, EDA-GA, ANN-BPSO and LBMBPSO methods have total cost of 52, 50.14, 48, 44 and 43 respectively. From the results the proposed HDDSC-JSO is effective and suitable for the cloud scheduling process. The load balancing algorithm of JSO, which balances load based on the weight allocated to each VM, provides VMs with a low load and high computational capacity for doing many tasks of varying sizes. The work with the greatest priority is assigned first to the VM with the highest weight. The suggested work's overall cost is reduced as a result of this method's increased VM efficiency in performing a variety of activities.

Degree of Imbalance performance comparisons

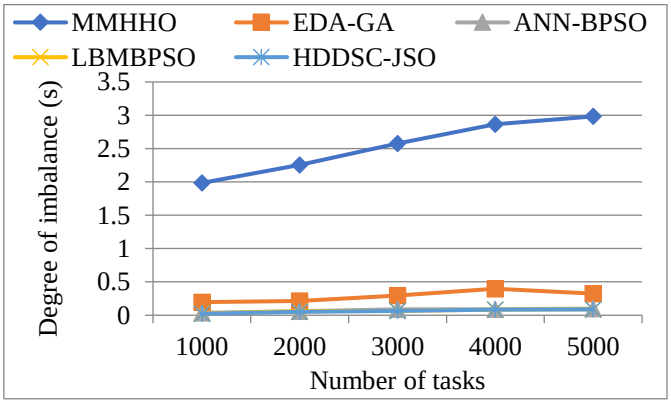

Fig.9. Degree of Imbalance comparison results

As demonstrated in Fig.9, The MMHHO, EDA-GA, ANN-BPSO, and LBMBPSO techniques are all more imbalanced than the LBMBPSO method. As a consequence, in terms of load balancing, the suggested

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 11 Issue 1, 2023
pp: 2111 - 2124

methodology performs better than the other present solutions. For 5000 tasks, the DI for the HDDSC-JSO method is 0.0874s while it is 2.983s, 0.324 s, 0.0998s and 0.0896s for MMHHO, EDA-GA, ANN-BPSO and LBMBPSO, 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. The suggested work, which is implemented by VM clustering, has a faster reaction time as a result than the present approach. Tasks are assigned and completed quickly, reducing the amount of imbalance, and VM migration is decreased.

CONCLUSION AND FUTURE WORK

This paper proposes using HDDSC and JSO algorithms to schedule and balance workloads to optimize user task response time. With the HDDSC algorithm, incoming tasks are grouped according to priority. Task scheduling clusters are created using priority values that are calculated based on the top and bottom levels of the tasks. The ADETT algorithm ranks scheduled tasks by rank value calculation, taking into consideration typical execution and communication durations. Next processes start with the highest priority task. Such task grouping and ranking improves resource consumption in this work when compared to that of previous approaches. The JSO approach helps load balance tasks with the system's existing resources while reducing makespan, DI, and cost. According to the simulation findings, the suggested work's performance metrics are superior to those of the current ones, especially in terms of reaction time reduction. The implementation of a powerful load balancing algorithm to speed up execution and the incorporation of a fresh, clever optimization technique to identify the work that needs to be given the greatest priority are both part of future plans. Deep learning algorithms may also be applied further in other scenarios. It is also possible to use additional machine learning algorithms. In order to prevent this waste of cloud resources, it was also intended to learn more about the dynamic load balancing technique. It was also recommended that steps be taken to decrease SLA breaches and improve service quality.

REFERENCES

1. Cusumano, M. (2010). Cloud computing and SaaS as new computing platforms. *Communications of the ACM*, *53*(4), 27-29.
2. Beimborn, D., Miletzki, T., & Wenzel, S. (2011). Platform as a service (PaaS). *Wirtschaftsinformatik*, *53*, 371-375.
3. Scheuner, J., & Leitner, P. (2019, March). Performance benchmarking of infrastructure-as-a-service (IaaS) clouds with Cloud Work Bench. In *Companion of the 2019 ACM/SPEC International Conference on Performance Engineering* (pp. 53-56).
4. Madni, S. H. H., Abd Latiff, M. S., & Coulibaly, Y. (2016). Resource scheduling for infrastructure as a service (IaaS) in cloud computing: Challenges and opportunities. *Journal of Network and Computer Applications*, *68*, 173-200.
5. Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: a toolkit for modelling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, *41*(1), 23-50.
6. Mauch, V., Kunze, M., & Hillenbrand, M. (2013). High performance cloud computing. *Future Generation Computer Systems*, *29*(6), 1408-1416.
7. Zhang, L., Zhou, L., & Salah, A. (2020). Efficient scientific workflow scheduling for deadline-constrained parallel tasks in cloud computing environments. *Information Sciences*, *531*, 31-46.
8. Liu, H. (2022). Research on cloud computing adaptive task scheduling based on ant colony algorithm. *Optik*, *258*, 168677.
9. Xie, Y., Sheng, Y., Qiu, M., & Gui, F. (2022). An adaptive decoding biased random key genetic algorithm for cloud workflow scheduling. *Engineering applications of artificial intelligence*, *112*, 104879.
10. Huang, X., Li, C., Chen, H., & An, D. (2020). Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies. *Cluster Computing*, *23*, 1137-1147.
11. Celik, E., & Dal, D. (2021). A novel simulated annealing-based optimization approach for cluster-based task scheduling. *Cluster Computing*, *24*(4), 2927-2956.

12. Aladwani, T. (2020). Types of task scheduling algorithms in cloud computing environment. *Scheduling Problems-New Applications and Trends*.
13. Murad, S. S., Badeel, R. O. Z. I. N., Salih, N., Alsandi, A., Faraj, R., Ahmed, A. R., ... & Alsandi, N. (2022). Optimized Min-Min task scheduling algorithm for scientific workflows in a cloud environment. *J. Theor. Appl. Inf. Technol*, *100*, 480-506.
14. Hung, T. C., Hieu, L. N., Hy, P. T., & Phi, N. X. (2019, January). MMSIA: improved max-min scheduling algorithm for load balancing on cloud computing. In *Proceedings of the 3rd International Conference on Machine Learning and Soft Computing* (pp. 60-64).
15. Ibrahim, I. M. (2021). Task scheduling algorithms in cloud computing: A review. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(4), 1041-1053.
16. Mangalampalli, S., Karri, G. R., & Kose, U. (2023). Multi Objective Trust aware task scheduling algorithm in cloud computing using Whale Optimization. *Journal of King Saud University-Computer and Information Sciences*.
17. Guo, X. (2021). Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm. *Alexandria Engineering Journal*, *60*(6), 5603-5609.
18. Talha, A., Bouayad, A., & Malki, M. O. C. (2022). An improved pathfinder algorithm using opposition-based learning for tasks scheduling in cloud environment. *Journal of Computational Science*, *64*, 101873.
19. Sharma, N., & Garg, P. (2022). Ant colony based optimization model for QoS-Based task scheduling in cloud computing environment. *Measurement: Sensors*, *24*, 100531.
20. Mahmoud, H., Thabet, M., Khafagy, M. H., & Omara, F. A. (2021). An efficient load balancing technique for task scheduling in heterogeneous cloud environment. *Cluster Computing*, *24*(4), 3405-3419.
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. Zhao, J., Yang, K., Wei, X., Ding, Y., Hu, L., & Xu, G. (2015). A heuristic clustering-based task deployment approach for load balancing using Bayes theorem in cloud environment. *IEEE Transactions on Parallel and Distributed Systems*, *27*(2), 305-316.
26. Chou, J. S., & Truong, D. N. (2021). A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean. *Applied Mathematics and Computation*, *389*, 125535.
27. 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.
28. 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.
29. 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.