Improving Load Balancing Efficiency in Cloud Data Centers Through Hybrid Grey Wolf with Cat Swarm Optimization

Arif Ullah[1], Siti Fatimah Abdul Razak[2*], Amine Mrhari[3], Sumendra Yogarayan [4], Syeda Hina Mazhar Kazmi[5]
[1]Centre for Intelligent Cloud Computing, Multimedia University, Melaka, 75450, Malaysia
[2,4]Faculty of Information Science and Technology, Multimedia University, Melaka, 75450, Malays
[3]Research in Computer Science Laboratory Faculty of Sciences, Ibn Tofail University Kenitra, Morocco
[5]Faculty of Information Science Arid Agricultural University Pakistan
E-mail: Arifullahms88@gmail.com & fatimah.razak@mmu.edu.my.
[*]Corresponding author

Keywords: virtual machine, makespan, throughput, degree of imbalance, turnaround time

Received: April 10, 2025

An effective resource management strategy that anticipates server resource utilization and appropriately distributes the load is recommended in order to address these problems and enhance data center performance. By reducing the number of servers in use, facilitating virtual machine migrations, and optimizing resource utilization, it helps save power. To reduce the likelihood of service level agreement (SLA) violations and performance degradation caused by either oveloded or under loaded servers and virtual machines. Resources for software applications can now be dynamically altered as needed thanks to the growth of cloud computing. Since better resource consumption can lead to increased scalability as well as significant cost and energy savings, effective resource management is crucial in cloud computing. The flexibility of cloud resources allows clients to dynamically increase and decrease their resource demands over time. However, predefined virtual machine sizes and variable resource requirements result in underutilization of resources, load imbalances, and high power consumption. The goal of this research is to develop a hybrid technique by combining Grey Wolf with algorithms. The hybridization processes take place in the Grey Wolf portion, when the Cat Swarm initialization process takes the place of the startup phase. The virtual machine (VM) section's data selection is enhanced by this substitution. The Grey Wolf and Cat Swarm algorithms are two examples of optimization algorithms. The evaluation criteria that are used are makespan, throughput, degree of imbalance, and turnaround time with degree of imbalance. The recommended approach outperforms alternative algorithms in each of these metrics. The proposed hybrid strategy resulted in 0.3% increase overall performance. Potential directions for future research include testing the proposed approach in larger and more complex data distribution in cloud data centers.

Povzetek: Predstavljen je izvirni hibridni algoritem HGWCA, ki združuje dva algoritma: Grey Wolf Optimizer in Cat Swarm Optimization za učinkovitejše razporejanje nalog in zmanjšanje porabe časa v oblačnih podatkovnih centrih.

1 Introduction

The term "cloud computing" refers to the provision of pay-as-you-go on-demand services via the internet. Instead of using a conventional computer system or any other local devices, it allows you to handle files online. The cloud is a network of interconnected virtual machines with distributed, parallel systems that may deliver and offer computing resources on demand. "Cloud computing" refers to anything that involves offering hosted services via the internet. Private, public, hybrid, and community clouds are the four types of cloud deployment models. Numerous cloud service providers are accessible, and each one offers a unique collection of cloud services. Platform as a Service (PaaS), Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Storage as a Service (STaaS), Cloud computing models that provide a range of services to customers include Security as a Service (SECaaS), Data as a Service (DaaS), Test Environment as a Service (TEaaS), and Backend as a Service (BaaS) [1]. Software as a Service (SaaS) offers software services online, with the service provider managing software maintenance and updates. IaaS enables virtualized access to actual resources, whereas PaaS offers tools for managing and developing applications. Instead of buying their own devices, businesses can rent storage capacity thanks to STaaS. Security services like intrusion detection and authentication are included in SECaaS. Regardless of one's location, DaaS makes it possible to access data whenever there is a need for it. Through the internet, TEaaS enables users to access software and pertinent data. Developers can concentrate on the frontend and application by using BaaS's pre-built backend services for mobile applications reasoning [2]. Cloud computing has a number of benefits. It gives users access to inexpensive systems, removing the need for powerful computers and enabling the use of less expensive gadgets. Because it makes data and apps accessible from any location at any time, it provides dependability and ease. Because they may use cloud computing instead of spending money on pricey servers, it also lowers the cost of IT infrastructure for businesses. There are less maintenance problems and expenses, which means that less hardware and software maintenance is needed. Users can always get the most recent version of software without extra expense or work because to cloud computing's smooth software upgrades. By utilizing the combined capabilities of linked PCs and servers, it provides more processing power [3]-[4]. Cloud storage surpasses the constraints of local storage by offering nearly infinite capacity. Usage of these resources by making certain that workloads are distributed fairly across the servers or virtual machines that are available. Better performance and resource efficiency in the cloud are ensured by load balancing, which helps avoid overloading certain servers by dynamically allocating and reallocating computing resources based on demand. In a cloud context, load balancing is the essential technology to guarantee a fair task distribution and effective resource use. One useful strategy for cloud data centers to save energy is to consolidate virtual machines (VMs). By combining virtual machines (VMs) into fewer active physical machines (PMs), the VM consolidation technique enables the PMs that do not have any VMs to go into a state of dormancy. Because a PM uses a lot of energy when it is sleeping, Utilizing VM consolidation techniques can assist cloud data centers consume less energy, as it is lower than that of a PM in its active mode [5]. To reduce waiting time, the virtual machine (VM) should react quickly when a tasks is sent to the cloud to be processed. However, tasks should be distributed among all virtual machines (VMs) in parallel to ensure system balance and efficient functioning making use of the available resources. This necessitates task planning that is assigned and distributed among the available resources. When several tasks are assigned to a single virtual machine (VM), the assigned activities will execute simultaneously across numerous VMs to complete the tasks. As a result, the task requirements ensure that not every tasks is loaded onto a single computer. The system is unbalanced, or the VM will restrict access to other VMs, rendering them unreachable [6]. Other factors, such as makespan, cost, and resource use, must be considered while scheduling in order to avoid this. The main goal of allocating among a system's load-balancing responsibilities is to maximize workload distribution among available resources and minimize system-processing timing. Several scholars have proposed methods for load balancing in both uniform and heterogeneous environments. In order to do this, the suggested algorithm in this study split the tasks evenly between VM and PM [7]. The structure of the paper is as follows: Section 2 presents the latest advancements in tasks distribution for hybrid systems for cloud systems. Section 3 describes the suggested model, and the algorithm created to address the issues and illustrate the design of our solutions and allocate resources. Section 4 describes the procedures, tests, and results and these conclusions are present in Section 5.

2 Related work

In this study, we examine the relevant literature on energy management, load balancing, and appropriate cloud data center use. In order to evaluate Quality of Service (QoS) metrics, a multi-objective task-scheduling algorithm that combines Cat Swarm Optimization (CSO) with machine learning classifiers such as Support Vector Machine (SVM) has been proposed. However, the algorithm falls short in terms of classification and consideration of other important parameters. Concepts like workload prediction and auto-scaling require further investigation [8]. A dynamic task scheduling system for tasks and cloudlets is suggested by the author [9]. The approach seeks to increase acceptance rate, scalability, and minimize makespan. However, it is expensive and complicated. Future studies will concentrate on load balancing strategies that respect privacy. As stated in [10], discuss the scheduling of cloud computing applications, with an emphasis on response speed and makespan. Enhancing these metrics and investigating the effectiveness of the MOTSGWO algorithm in an open stack setting are their goals. The Sand Cat Swarm Optimization (SCSO) algorithm for global optimization problems is introduced in [11]. Although the technique can be expensive, it works effectively in a variety of engineering design challenges. Additionally, it may find use in bioinformatics, machine learning, smart farming, logistics, wireless sensor networks, and the Internet of Things. For feature selection in classification tasks, a binary multi-objective grey wolf optimizer is proposed in [12]. With a lower computational cost, the algorithm performs better than current methods in terms of features reduction and classification error rate. It works well with small datasets introduces a clustering technique based on RCSO that makes use of SVM and MLP networks. The RCSO algorithm performs better than current algorithms and may be applied in future tissue cell quantification and legal proceedings. An approach for the Urban Transit Routing Problem (UTRP) based on Cat Swarm Optimization (CSO) is presented in [13]. It is economical and successfully resolves the UTRP. However, the values of the parameters affect the algorithm's capacity to optimize. A Cat Swarm Optimization (CSO) algorithm using a subgroup information interaction approach is proposed by the author in [14]. It exhibits a faster rate of convergence and the capacity to search globally. It will be compared to other swarm intelligence algorithms in future studies. A Sand Cat Swarm Optimization (SCSO) algorithm based on elite collaboration and [15] present stochastic variation. Although it increases time complexity, it effectively addresses engineering and optimization difficulties. For more applications, cooperation with other researchers is desired. As suggested by [16], a MATLAB task scheduling system that reduces makespan and energy usage. Energy and execution costs, however, are not taken into account. Evaluation of the suggested algorithm in a small-scale cluster is the goal of future research. The enhanced Multi-Objective Grey Wolf Optimizer (IMOGWO) algorithm for edge computing work scheduling is presented in paper

[17]. In terms of convergence, diversity, and coverage, IMOGWO performs better than current algorithms. The algorithm can be used for multi-objective test issues; however, it is expensive suggests using the Hybrid Particle Swarm Grey Wolf (HPSGW) algorithm to optimize automated CNN in [18]. The technique lowers computational costs while increasing performance accuracy on benchmark datasets. Future studies will concentrate on enhancing the optimization procedure by adding more hyperactive parameters. A hybrid form of MCA and GWO for scheduling the power of smart home appliances is presented by [19]. The Power scheduling issues are successfully addressed by the GWO-MCA hybrid. The goal of future study is to enhance the selection process and method's performance. For multi-objective optimization problems, [20] suggests a method based on the Grey Wolf Optimizer (GWO) with the memeplex structure of the Shuffled Frog leaping method (SFLA). The method can be used to solve a variety of problems and produces competitive results. In order to optimize spectrum sensing in cognitive radio networks, [21] presents the Multi-Objective Modified Grey Wolf Optimization (MOMGWO) algorithm. In spectrum sensing, MOMGWO has excellent coverage, convergence, and performance. The Multi-Objective Grey Wolf Optimizer (MOGWO) method is suggested by [22] for the IEEE-30 bus test system. For multi-objective problems, MOGWO shows good Pareto-optimal front setup and quick convergence. [23] Introduce the Fractional Grey Wolf Optimizer (FGMTS). For cloud Computing multi-objective tasks scheduling. In addition to achieving resource allocation, the FGMTS method enhances cloud profitability and performance. However, because of the search space, there can be restrictions. The multi-objective Grey Wolf Optimizer (MOGWO) for near-optimal tasks scheduling in cloud computing is presented in paper [24]. The MOGWO performs better than alternative approaches and can be improved with parallel programming techniques. The summary of related work is shown in Table 1.

Table 1: Related work summary

Ref	Methods	Parameter	Results	Limitations	Future work
[25]	Dynamic task scheduling	Makespan	Well as improving scalability	Algorithms are expensive and complicated	Effectively safeguard sensitive task
[26]	Multi objective grey wolf optimizer (MOGWO)	Convergence	Fastest convergence and the best Pareto-optimal front setting. Reduce fuel cost	Laborious and costly	It is possible to obtain a Pareto-optimal front for multi-objective
[27]	FGMTS: Fractional grey wolf optimizer	Multi-Objective task scheduling	Performance and increasing profitability.	Result in faulty resource allocation for subtask	Could be used for other datasets
[28]	Task scheduling mechanism	Makespan and energy consumption	It greatly minimizes energy consumption and makespan	Energy and execution cost are not considered	Assess the proposed algorithm while taking into account a small-scale cluster.
[29]	IMOGWO algorithm	Task Scheduling	Convergence, diversity and coverage.	Costly	Applied to multi-objective test problem
[30]	An algorithm based on grey wolf optimizer (GWO) with memeplex structure of the shuffled frog leaping algorithm (SFLA)	Multi objective optimization	Multi-objective problems and achieve competitive or superior results when compared to other comparison algorithms.	Complicated	It can also be tested on multi-objective types of real-world problems such as image processing, vehicle routing and clustering.
[31]	Cat Swarm Optimization (CSO)-based algorithm	Optimization, scheduling workflow	CSO based algorithms are an excellent choice for effectively solving the UTRP while also being cost-effective.	The determination of parameter values, which may be lacking in effectiveness.	The algorithm aims to optimize the cost of the service provider while also being applicable
[32]	A binary multi-objective grey wolf optimizer for feature selection in classification	Features number, classification accuracy	Classification error rate though benefiting from a lower computational cost.	Effective for small size datasets	Minimizing the number of features and minimizing the error rate simultaneously
[33]	Multi objective grey wolf optimizer (MOGWO)	Throughput, makespan and resource utilization in cloud computing.	The method outperformed all other methods tested	Complicated	Memory usage during peak loads, to improve overall efficiency.

The main goal of dynamic task scheduling systems is to decrease makespan while improving system scalability; however, their algorithms are frequently costly, intricate, and made to properly protect sensitive activities. Although it is still time-consuming and expensive to install, the Multi-Objective Grey Wolf Optimizer (MOGWO) is known for reaching the fastest convergence rates and providing the best Pareto-optimal front settings, with advantages including fuel cost savings. These capabilities are extended for multi-objective task scheduling by the

FGMTS (Fractional Grey Wolf Optimizer), which enhances system performance and profitability. However, it can occasionally lead to incorrect subtask resource allocation, which may be applicable to other datasets. However, they frequently ignore the combined effect of energy and execution costs, traditional task scheduling techniques focus on decreasing makespan and energy consumption, restricting their assessment to small-scale clusters. By striking a balance between convergence, diversity, and coverage, the IMOGWO method significantly improves task scheduling. However, because of its high computational costs, it is more appropriate for controlled multi-objective test problems. Another version combines the Grey Wolf Optimizer with the Shuffled Frog Leaping Algorithm's (SFLA) structure. It produces competitive results in multi-objective optimization, yet it is still sophisticated and appropriate for applications such as clustering, image processing, and vehicle routing. Targets for feature selection using a binary multi-objective GWO although it works best with small datasets, it can achieve low error rates and lower computing costs by decreasing the number of features and increasing classification accuracy. Furthermore, throughput, makespan, and resource utilization are improved by a multi-objective GWO version for cloud computing environments, which continuously outperforms alternative approaches but struggles with complexity and excessive memory use during periods of high load. Overall, these techniques tackle different facets of multi-objective scheduling and optimization problems in cloud and workflow contexts; each has particular advantages and disadvantages.

2.1 Problems statement

The Grey Wolf exploitation strengths are enhanced by the Cat Algorithm's exploration skills, resulting in a synergistic optimization framework. This hybrid approach seeks to allocate workloads across available resources in a dynamic manner while tackling several goals, including throughput, energy efficiency, and response speed. In a fast changing technological world, the incorporation of these bio-inspired algorithms into load balancing offers a fresh way to get around the drawbacks of current methods, improving the scalability and performance of cloud data centers. The Cat Algorithm's exploration abilities complement the Grey Wolf exploitation capabilities, creating a synergistic optimization framework. This hybrid strategy aims to dynamically distribute workloads across available resources while addressing a number of objectives, such as response speed, energy efficiency, and throughput. The integration of these bio-inspired algorithms into load balancing provides a novel approach to overcome the limitations of existing techniques, enhancing the scalability and performance of cloud data centers in a rapidly evolving technological environment.

3 Proposed algorithm

A task manager, scheduler, CPU, RAM, storage resources, virtualization layer, physical resources, and a data center are among the parts of the system shown in Figure 1.

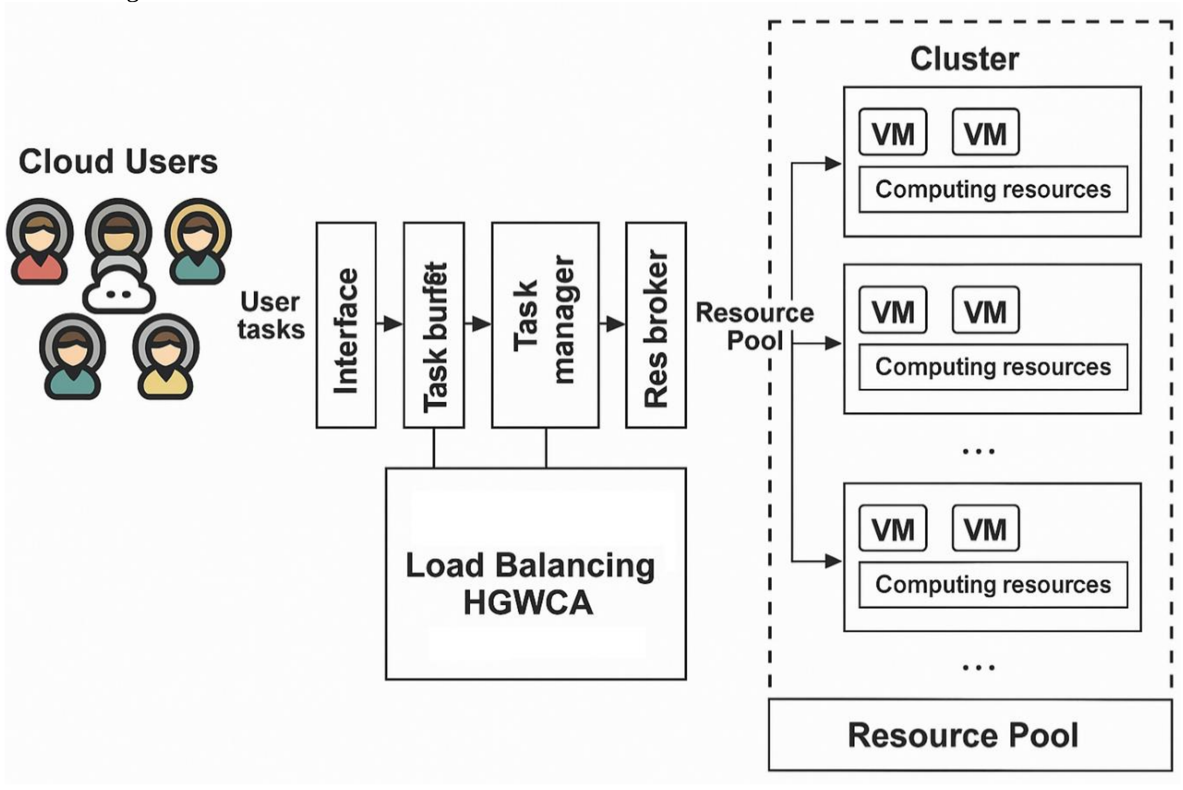

Figure 1: Proposed approach model

The grey wolf optimization algorithm (GWO) is a new type of metaheuristic algorithm. In the field of engineering, it is most commonly used to tackle optimization problems. It is inspired by the hunting and social behaviors of grey wolves. It imitates the natural

Leadership structure and hunting technique of grey wolves. Grey wolves do most of their living in packs. In a pack of grey wolves, the leadership hierarchy is indicated by the four categories of alpha, beta, delta, and omega. Each category denotes a different degree of authority and leadership within the pack. The dominant member of the pack, the alpha wolf, acts as a leader to guide the pack through the hunting process. Beta wolves [34] occupy the second level of this hierarchy. They are known as subordinates and assist. The alpha when creating policies during the hunting phase. After the alpha and beta wolves, the delta wolf represents the third level of decision-making authority. The last rung in this ladder is omega wolf. That is what scapegoat wolves are. Grey wolves' decision-making skills decline in the absence of the dominating alpha wolf. The gray wolf Optimizer is mostly inspired by this social skill [35]. Grey wolves hunt in three repeating stages utilizing a methodical approach. These phases include searching, encircling, and assaulting. This first phase is called tracking or searching. It is also known as the exploration stage. In a global search arena, the grey wolves look for prey. The mathematical the model describes how to determine the distance between the grey wolf and its prey [36]. It is computed using Equation 1.

$$Dist = abs(coeff \times L_P\ (t) - L_{GW}\ (t)) \quad (1)$$

Here $coeff$ provides the coefficient vector; $L_{GW}\ (t)$ shows where the grey wolf was in the specified time interval t; L_P shows the location of prey at a specified time interval t.

The value of $coeff$ is calculated using Equation 2.

$$Coeff = 2 \times Ran(0,1) \quad (2)$$

Due to the potential for random task generation and the consequent random assignment of tasks to the seeking and tracing modes, n has been randomly set as the number of dimensions. At each iteration, solutions are computed for cats, and their fitness is evaluated relative to other cats, until the optimal solution is found. While in the "seeking" mode, all cats will enter a resting state, however specific behaviors are required to find a cat. The new velocity in the tracing mode can be computed using the cats' observations and the known initial velocity [37]. You may see it below. Due to the potential for random task generation and the consequent random assignment of tasks to the seeking and tracing modes, n has been randomly set as the number of dimensions. At each iteration, solutions are computed for cats, and their fitness is evaluated relative to other cats, until the optimal solution is found. While in the "seeking" mode, all cats will enter a resting state, however specific behaviors are required to find a cat. The new velocity in the tracing mode can be computed using the cats' observations and the known initial velocity, you may see it below [38].

$$Vel_i^d(t+1) = p * Vel_i^d(t) + c * r * (x_d^{best} - x_i^d) \quad (3)$$

In Equation (3), c is a constant, r is a random value between 0 and 1, and $Vel_i^d(t)$ is the velocity of the ith at during the t-th iteration. The cat is on its top place globally atx_d^{best}. Cat location at ith iteration is represented byx_i^d.

By using the Equation (4), the location of the cat may be modified as follows.

$$x_i^d(t+1) = x_i^d + Vel_i^d(t+1) \quad (4)$$

Following fitness calculations, solutions must be updated with non-dominated cats. These calculations must be carried out until cats achieve objectives with an optimal value. The termination criterion is used to end the algorithm. For the algorithm to work well, a suitable termination condition needs to be selected. The number of iterations, the degree of improvement, and the running time are typical CSO termination criteria [39]. During the third phase, grey wolves adjust their locations based on the movements of their prey, as shown in equations (1) through (6). In the grey wolf, social structure, alpha, beta, and delta wolves hold the highest positions and are thought to be the finest hunting alternatives. After the prey has been surrounded at a specific distance, it is believed that the alpha, beta, and delta wolves have a better idea of its whereabouts. Consequently, the alpha wolf initiates the attack. The prey's position is subsequently updated using the alpha, beta, and delta wolves' positions, as mathematically demonstrated in equation (5). In contrast, the omega wolves follow the guidelines in equation [40] and move around the prey intermittently.

$$\begin{gathered}
Dist_{alpha=}\ abs(coeff_1 \times L_{alpha} \times (t) - L(t)) \\
Dist_{beta=}\ abs(coeff_2 \times L_{beta} \times (t) - L(t)) \\
Dist_{delta=}\ abs(coeff_3 \times L_{delta} \times (t) - L(t)) \\
L_1 = L_{alpha}(t) - coeff_1 \times Dist_{alpha} \\
L_2 = L_{beta}(t) - coeff_2 \times Dist_{beta} \\
L_3 = L_{delta}(t) - coeff_3 \times Dist_{delta} \\
L = \frac{(L_1 + L_2 + L_3)}{3} \quad (5)
\end{gathered}$$

A set of random solutions are used as grey wolves by the grey wolf optimizer. Several objective functions are used to evaluate a set of randomly selected responses. The quality of each solution is represented by the values of a fitness function composed of many functions. Alpha, beta, and delta wolves stand for the highest three quality solutions. The grey wolf optimizer continuously adjusts the wolf population's location throughout each cycle. If a response improves during an iteration to the extent where it outperforms alpha, beta, and delta wolves, the corresponding solution is replaced with the new one. When a set of stopping conditions is met, the Grey Wolf Optimizer stops iterating the solutions [41]. While the scheduler distributes resources like CPU, memory, and storage, the task manager organizes tasks. To manage virtual machine or networks, the system has a virtualization layer and virtualized resources. The infrastructure is kept in the data center, and the physical resources stand in for the underlying hardware. Additionally, Figure 1 mentions the HGWCA algorithm, which is suggested to be involved in resource manageme nt. Tasks are initially considered TK= {T1+T2+T3… Tn}. The cloud console must be used to submit these tasks to the task manager first. The task management module in this configuration needs to assess each task's relative priority [42]. Task, and then assess each virtual machine's relative significance based on how much power it uses in a certain amount of time. It is being tried because there are many different types of tasks being brought into the cloud platform, and it is crucial to map them to the appropriate virtual resources in the cloud. The task scheduler is in charge of carrying out this task. A resource manager module, which is connected to a task scheduler, keeps tra ck of resource requests, allocations, and availability on th e respective physical hosts in the datacenters [43]. Assum ed in this method are n virtual machines (VMs) with nam es such as these virtual machines must live in physical hosts with names like Vn={V1,V2,V3,……Vn }.

$H_i = \{H_1, H_2, H_3, \dots \dots H_i\}$

In addition, these hosts are housed in datacenters (called centers) with names like

$$d_j = \{d_1, d_2, d_3, \dots \dots d_j\}. \qquad (6)$$

The architecture in question assigns tasks to virtual machines (VMs) by determining the Task Priority and VM Priority, respectively, while considering the datacenters' relative power costs [44]. To allocate each task to the best virtual machine (VM), a priority calculation must be performed because every task has different processing needs. The cloud's virtual machine (VM) priority computations must take regional variations in power prices into consideration because it can be deployed anywhere [45]. A technique that first identifies the most critical task and then allocates them to virtual machines that are most suited to completing those tasks with the least amount of energy is used in cloud data centers to reduce overall power costs and energy usage. Tasks, virtual machines, real hosts, and datacenters can all be used to define the problem. The scheduler's purpose is to distribute workloads to virtual machines in a way that minimizes costs and power consumption [46]. The given equation 7 present the hybrid section of the proposed algorithm.

$$Vi(0) = \{\, X\ i(current) + \delta \times R, X\ i(current) + r \times (X\ best - X\ i(current) \qquad (7)$$

It made a number of assumptions regarding datacentersd_j, physical hostsH_i, virtual machines V_n and tasks T_K in order to schedule tasks onto virtual machines. It must ascertain the relative significance of virtual machines and tasks for this configuration. These settings are examined and given a single priority when tasks are submitted to the task manager. The scheduler assigns tasks to the VMs, and in order to determine priorities, it is necessary to ascertain the VMs' present load. The VMs perform the tasks that the scheduler allocates, requiring the determination of their current load to calculate priorities. The equation below demonstrates the calculation of the load on all VMs [47].

$$L_v = \sum_{n=1}^{q} L_n \qquad (8)$$

The current load on each virtual machine (VM) is represented by the variable L_n since all of the virtual machines are meant to be housed within physical hosts; the capacity of the hosts must be ascertained after the load on each VM has been computed.

$$L_h = \frac{L_v}{\sum_{k=1}^{m} H_k} \qquad (9)$$

The physical host, which is composed of virtual machines, is under L_h load. Because the VMs must transfer to the next Virtual Machine on the same physical host either by starting a new request or to the subsequent VM that is already running if there are more tasks than they can handle, a load-balancing module is necessary in the cloud-computing paradigm. A load balancer can accomplish this, but in order to do so, a specific threshold value must be set to identify if the system is balanced or not. The system's threshold value is identified as follows [48].

$$TH_k = L_h * H_K \qquad (10)$$

Prior to determining whether the system is overloaded, under loaded, or balanced, the threshold value is established. The load balance is calculated to arrive at this conclusion. In the event that a system gets overwhelmed.

$V > TH_k - \sum_{i=1}^{n} L_V$

If the system is under loaded, then

$V < TH_k - \sum_{i=1}^{n} L_V$

If the system is balanced,

$$V = TH_k - \sum_{i=1}^{n} L_V \qquad (11)$$

To plan work onto appropriate VMs, it is necessary to see the processing capabilities of each VM. The calculation is as follows [49].

$$Proccapicity_V = pr^{no} * pr^{mips} \qquad (12)$$

The above equation is used to calculate each VM's processing capacity, and the results are listed as follows.

$$TotProccapicity_V = \sum_{i=1}^{n} Procapacity_V \qquad (13)$$

The requirements for allocating task to virtual machines have been determined and computed. The priorities for both the tasks and virtual machines have now been determined. The aforementioned formula has been used to calculate the work's size, and the relationship between a task's size and a virtual machine's processing power is what defines its priority. The following is the formula [50].

$$T^{priority} = \frac{T_k^{size}}{Proccapicity_V} \quad (14)$$

Two stages of verification are used to schedule the tasks in this work. In order to accurately map tasks onto the appropriate virtual machines (VMs), the first step entails determining task priorities. To make the mapping process easier, VM priorities are then established according to power unit cost. The following algorithm based on power unit cost determines VM priority [51].

$$V^{priority} = \frac{elec\ unit\ cost^{\ high}}{elec\ unit\ cost\ _j^d} \quad (15)$$

The metrics this tasks address must be identified after the priorities have been calculated. Any cloud computing task-scheduling system must consider makespan because it is a crucial metric to manage in this paradigm. Additionally, lowering response time has a significant positive impact on cloud providers and users, which in turn has an indirect impact on other metrics like energy consumption and total power cost. Here is the formula [52].

$$m^k = availability^n + e^k \quad (16)$$

4 Experiments and results

An analysis of the various experimental outcomes obtained by applying the recommended methodology is presented in this section. A wide range of datasets and parameters has been used in numerous experiments and testing. A thorough explanation of the study's experimental setup provides crucial context for understanding the methodology. The general process employed in the tests is described through a number of precisely specified points. This section provides comprehensive documentation of the experimental details associated with the proposed model. The primary objective of this section is to outline the exact configuration that was utilized to conduct the experiment as well as the systematic procedures that were followed in order to execute the recommended strategy within the experimental inquiry. As a result, the experimental setup and specifics will be covered in detail in the parts that follow.

4.1 Dataset

The availability of appropriate datasets, which were taken from the AWS (DREAM) database for this study, is a crucial component for the scheme's effective implementation. The Proposed model performance evaluation included a range of virtual machines (VMs) and workloads (between 200 and 2000) spread across various data centers in the cloud-computing environment.

4.2 Simulation parameters

A modified CloudSim 3.0 platform was used to construct the HGWCA algorithm, and test datasets with file sizes between 200 and 400 kilobytes and compliance with the standard workload format (SWF) were used. The simulation parameters are shown in Table 2.

Table 2: Simulation parameter

Type	Parameter	Value	Type	Parameter	Value
Region	From 1 to 4	5	Task/Data	No. of task	100/1600
Data center	No. of data center	5		Length of task	100/200/400 byte
	No. of hosts	100/1000		No. of processor per requirement	250 KB
	Type of manager	Time and space		Type of manager	Time and space
	Bandwidth	1000	Memory	Total memory	204,800 MB
Virtual Machine	Total no. of VM	30/20/20		No. of processor	4 per VM
	No. of processor per VM	4		Total processor	120
	VM memory	512		Storage memory	100000 Mb
	Bandwidth in bit	1000		Viable memory	10,000
	VM image size	1000	Cloudlets	Total no. of task	100/2000

A variety of parameters and their corresponding values pertaining to various system features are included in Table 2. It contains details about tasks, data centers, regions, virtual machine specs, algorithms, and dataset properties. The number of data centers, task duration, manager type, bandwidth, virtual machine configurations, CPU specs, and dataset specifics are a few examples. A thorough summary of these factors is given in Table 2, which also offers information on the setup and features of the system.

4.3 Performance metrics

With an emphasis on important metrics of load balancing effectiveness such makespan, throughput, turnaround time, and degree of imbalance, the proposed model performance was evaluated against that of other well-known algorithms, including ABC, MBat, HHO-ACO, and QMPSO. The following assessment measures are employed to examine the performance. These measures are described are as: The time required to process every tasks in the designated order is known as the make span. The study's primary goal is to reduce the makespan time. Makespan can be computed using the following formula:

$$Makespan = \left(\frac{Given\ time\ of\ network}{Complete\ time\ of\ network}\right) * 100 \quad (17)$$

Throughput is the quantity of data that can be sent and received in a given period. Throughput is the average ratio at which messages are successfully delivered to their intended location. The provided formula [53] can be used to compute it.

$$Throughput = \frac{No.of\ Tasks}{makespan} \quad (18)$$

The rate at which tasks are finished in a specified amount of time is referred to as throughput. It shows how much work is completed or processed in a certain amount of time. "Number of Tasks," the numerator of the equation, is the total number of tasks finished in the allotted time. It shows how much work or tasks have been completed successfully. The degree of imbalance quantifies the haphazard distribution of cloud workloads among virtual machines based on their respective capacities. It is usually computed using virtual machine task execution times. The provided formula [54] can be used to calculate it.

$$Imbalance\_Degree = \frac{\text{Max}\ \_CTime_i - Min\ \_CTime_i}{Avg\ \_CTime_i} \quad (19)$$

Here, Max $\_CTime_i$ denotes the maximum completion time of tasks on all virtual machines. $Min\ \_CTime_i$ Denotes the minimum completion time of tasks on all virtual machines.$Avg\ \_CTime_i$ Denotes the minimum completion time of tasks on all virtual machines. Imbalance Degree is a measure of how well load balancing is working in the cloud. Lower values show that Workload on the cloud is properly balanced. While higher values demonstrate inefficient, load balancing [55].

5 Results and discussions

With an emphasis on important metrics of load balancing effectiveness such makespan, throughput, turnaround time, and degree of imbalance, the proposed model performance was evaluated against that of other well-known algorithms, including PSO, ABC, PSO-CALBA, M-Bat, HHO-ACO, Proposed model. In comparison to other algorithms, the suggested Proposed modeled menstruated an impressive 0.98% decrease in makespan and a higher accuracy rate of .18%, according to the comparative analysis. These results validate the feasibility and efficacy of the proposed model in resolving load balancing issues in cloud computing settings and the result details are given below table 3 present the throughput.

Table 3: Complete no. of task of different algorithm based on No. of Task

Given No of Task	PSO	ABC	PSO-CALBA	M-Bat	HHO-ACO	Proposed model
200	160	165	170	175	186	193
400	355	365	372	378	385	390
600	560	570	575	580	588	592
800	750	770	775	780	786	791

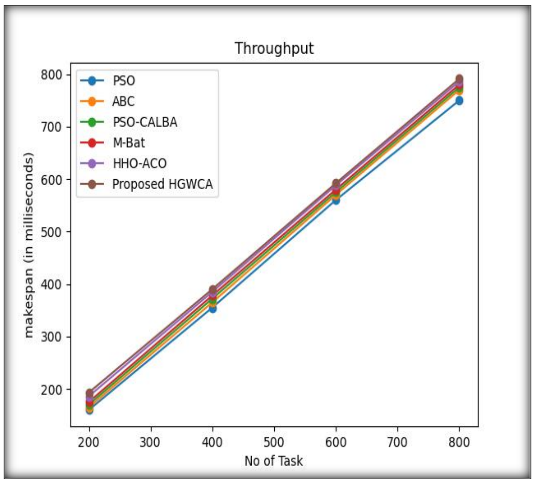

Figure 2: Throughput

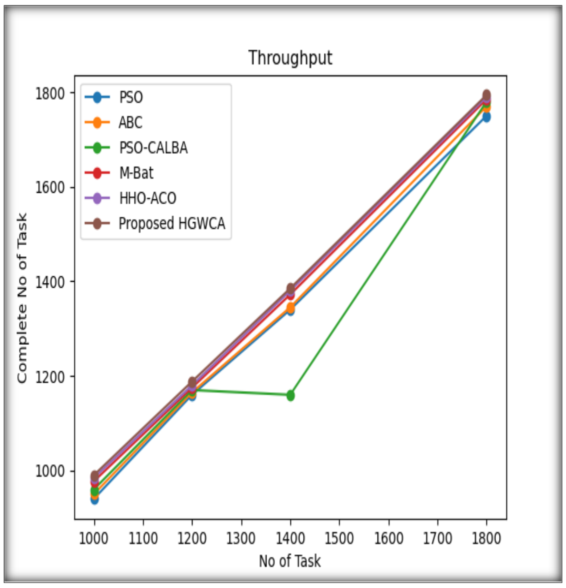

Figure 3: Throughput

Table 3 and Figure 2 show that the "Proposed model" algorithm performs the best based on the provided table. In a variety of task counts, it regularly receives the highest marks. With 800 tasks, it has a throughput of 791. In contrast, the "PSO" algorithm performs the poorest in the table provided. In a variety of task counts, it regularly receives the lowest scores. With 800 tasks, it has a throughput of 750.Thus, of the stated algorithms, the Proposed model method performs the best based on the data presented, while the PSO algorithm performs the poorest. The "Proposed model" method is clearly the best-performing algorithm, according to the statistics shown in Figure 3. With a throughput of 1794 when the number of tasks is 1800, it continuously obtains the highest marks across a range of task counts. On the other hand, in the provided table, the "PSO" algorithm performs the worst. With a throughput of 990 when there are 1800 tasks, it consistently receives the lowest scores throughout a range of task counts. Therefore, it can be inferred from the information given that, among the algorithms indicated, the proposed model method performs the best, while the PSO algorithm performs the worst. Figure 4 and Table 4 present the Makespan of the proposed algorithm.

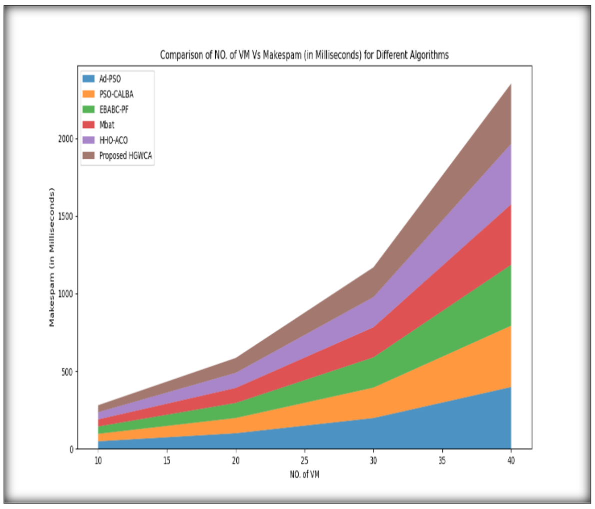

Figure 4: Makespan

Table 4: Makespan of different algorithm based on no. of virtual machine

No. of VM PSO	ABC	PSO-CALBA	M-Bat	HHO-ACO	Proposed model
10	48	48	46	46	45
20	100	98	97	97	96
30	198	196	194	194	192
40	398	395	390	390	388

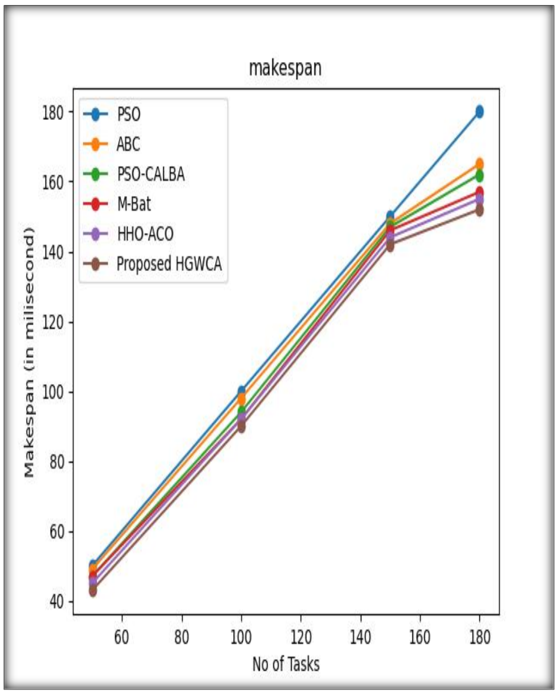

Figure 5: Comparison no. of VM Vs no task for different algorithms

The "Proposed model" technique is the best one based on the data presented in Table 5 and Figure 4, as it consistently produces the lowest makespan values for varying numbers of virtual machines. With consistently greater makespan values than other algorithms, the "PSO" algorithm is the worst. This suggests that when it comes to reducing execution time, the "Proposed model" algorithm outperforms the "PSO" approach.

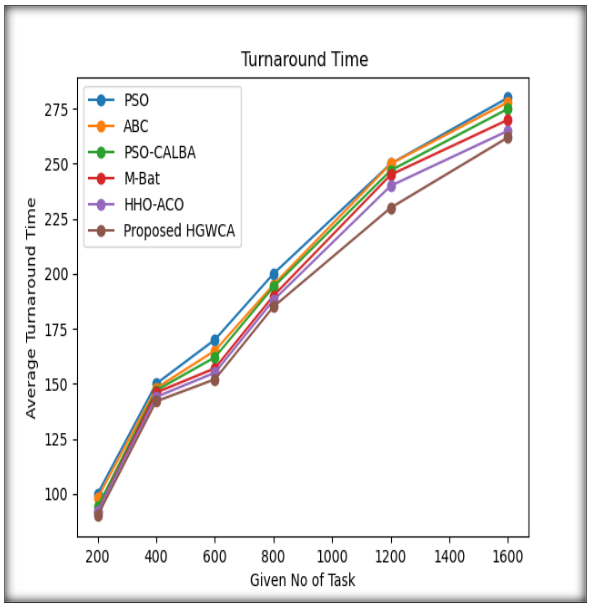

Table 6: Makespan based on no. of task (50-180)

Data on the number of completed tasks and the corresponding makespan (in milliseconds) for various algorithms are shown in Table 5 and Figure 5 and 6. The "Proposed model" method consistently obtains the lowest makespan values for all sets of full tasks, based on the data provided. As a result, it can be regarded as the optimal method for reducing execution time.

Table 5: Makespan based on no. of task

Algorithm Name	Complete No of Tasks	Makespan (In Milli second)	Complete No of Tasks	Makespan (In Milli second)	Complete No of Tasks	Makespan (In Milli second)	Complete No of Tasks	Makespan (In Milli second)
PSO	200	200	220	220	250	250	280	280
ABC	200	195	220	218	250	250	280	278
PSO-CALBA	200	194	220	215	250	247	280	275
M-Bat	200	190	220	210	250	245	280	270
HHO-ACO	200	188	220	207	250	240	280	265
HGWCA	200	185	220	202	250	230	280	262

Table 5 displays, for a range of parameters, the average results produced by numerous writers, including the suggested method. For varying task amounts, the authors' algorithms—PSO, ABC, PSO-CALBA, M-Bat, HHO, and the suggested approach—are assessed.

The following are the average results for each algorithm across various job quantities: M-Bat (169.63), HHO (167), PSO (177.50), ABC (175.13), PSO-CALBA (172.63), and the suggested technique (163.25). With lower numbers signifying better results, these average values show how well each method performs in relation to the given parameter.

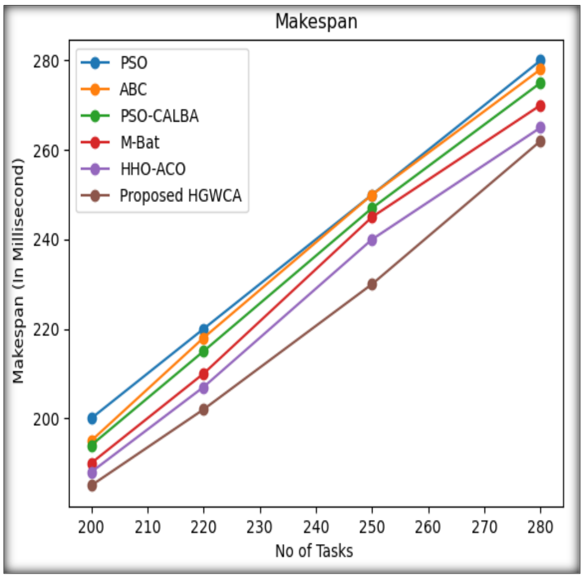

Figure 7: Makespan based on no. of task (200-280)

Data on the number of finished tasks and the accompanying makespan (in milliseconds) for various algorithms are shown in Figure 7 and Figure 8. Across all sets of finished tasks, the "Proposed model" algorithm consistently obtains the lowest makespan values, demonstrating economical execution time. The "PSO" method, on the other hand, continuously displays increasing makespan numbers, suggesting a comparatively slower execution time. According to this data, the "PSO" method performs relatively worse than the "Proposed model" approach in terms of minimizing execution time. The average turnaround time of several algorithms for varying amounts of tasks is shown in Figure 6 and Table 6. Across all task sets, the "Proposed model" algorithm consistently shows the lowest average turnaround time, suggesting effective completion. The "PSO" method, on the other hand, typically shows greater average turnaround time values, suggesting comparatively slower task completion. According to these findings, the "Proposed model" algorithm does exceptionally well in reducing average turnaround time, whereas the "PSO" method typically performs worse in this area.

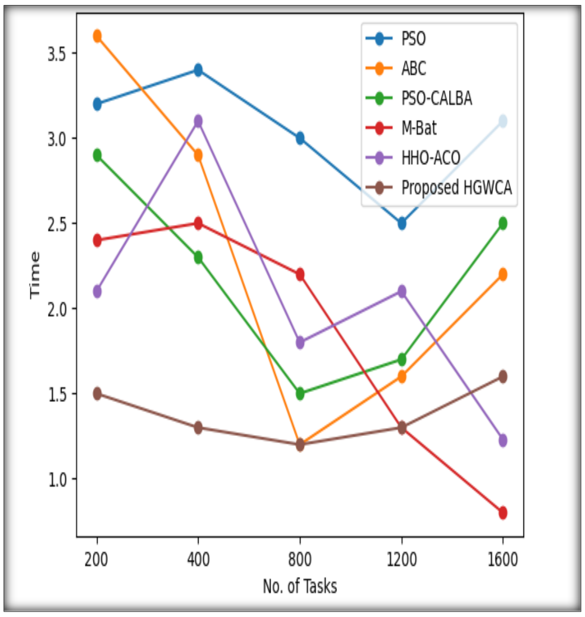

Figure 8: Turnaround time of tasks

Data on cloudlet turnaround times for different algorithms and task quantities are included in Table 6 and Figure 8. The "Proposed model" algorithm's efficiency in finishing jobs is demonstrated by the fact that it consistently displays the quickest turnaround time values for all task sets. The "M-Bat" algorithm performs competitively in contrast, whereas the "PSO" and "HHO-ACO" algorithms typically have greater turnaround times. The "Proposed model" algorithm is the most effective at reducing cloudlet turnover time overall.

Table 6: Throughput time of tasks

No. of Task	200	400	800	1200	1600
PSO	3.5	5.8	6.5	5.1	6.6
ABC	2.2	2.4	2.1	2.6	2.9
PSO-CALBA	3.1	4.2	6.7	6.8	8.6
M-Bat	2.9	2.8	2.8	2.2	3.1
HHO-ACO	0.67	0.45	0.58	0.99	0.23
Proposed model	3.4	5.9	6.6	8.1	7.2

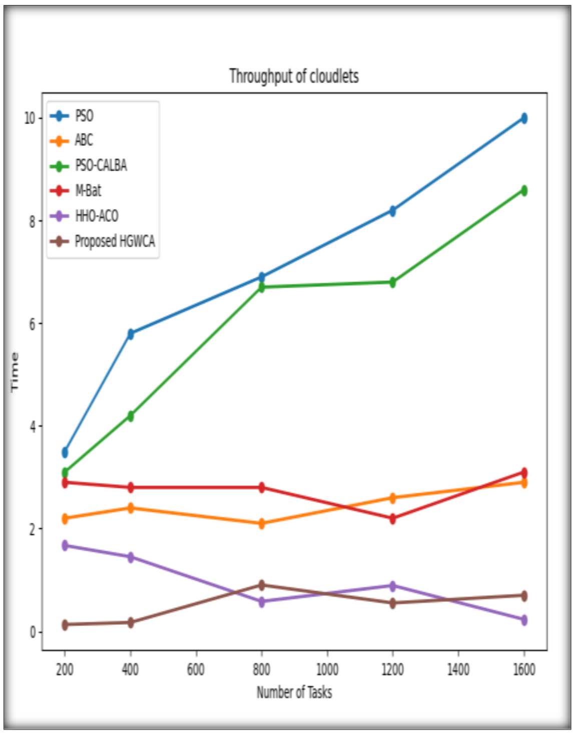

Figure 9: Throughput time of tasks

Table 7 and Figure 9 analysis show that the algorithms' throughput times vary depending on the work quantity. Across all task sets, the "HHO-ACO" algorithm consistently shows the lowest throughput time values, demonstrating its effectiveness in swiftly processing cloudlets. On the other hand, when compared to other algorithms, the "PSO-CALBA" algorithm typically displays higher throughput times, indicating slower processing. Across a range of task amounts, the "ABC" and "M-Bat" algorithms exhibit competitive performance with comparatively shorter throughput times. Although the "Proposed model" algorithm's throughput time numbers vary based on the number of tasks, it works reasonably well.

Table 7: Degree of imbalance of cloudlets

No. of Task	PSO	ABC	PSO-CALBA	M-Bat	HHO-ACO	Proposed model
200	2.8	2.6	2.9	2.4	0.66	0.87
400	3.4	2.9	2.3	2.5	1.2	0.8
800	2.7	1.2	1.5	2.2	0.9	0.5
1200	3	1.8	0.7	1.3	0.31	0.39
1600	3.1	2.2	2.5	0.8	0.21	0.19

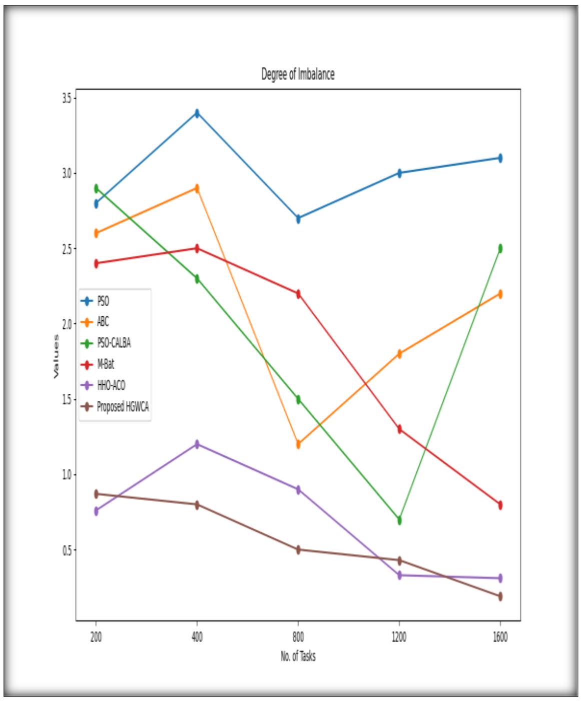

Figure 10: Degree of imbalance

It seems that the "Proposed model" algorithm does good tasks at attaining a decreased degree of imbalance across different task quantities, based on the numbers in Table 7 and Figure 10. Table 2 to 7 and Figure 4 to10 demonstrate that, in comparison to the other methods on the list, the suggested approach consistently produces the lowest makespan value. The makespan is the amount of time needed to do every task in a scheduling problem. The algorithm is thought to be more efficient the lower the makespan. The superiority of the suggested algorithm holds true for tasks varying in size from 100 to 2000. This illustrates how resilient and dependable it is when managing different workload scenarios. It is regarded as the ideal option because it may consistently produce the lowest makespan value, demonstrating the best possible resource utilization and tasks scheduling.

5 Conclusions and future work

In conclusion, using the Grey Wolf Optimizer (GWO) and Cat Swarm Optimization (CSO) algorithms to boost makespan and throughput in cloud data centers has produced promising results. It has been shown that using these ingenious methods can optimize resource allocation and task scheduling, resulting in a reduction of makespan and an increase in throughput. The study's findings show how effectively GWO and CSO investigate and make use of the solution space while accounting for several cloud-specific characteristics.

Through clever resource allocation and job scheduling, these algorithms may enhance cloud data center performance, leading to higher customer satisfaction and productivity. Further research in this area is necessary to improve optimization techniques and sustain continuous cloud advancement data center operations. Restrictions In order to improve makespan in cloud data centers, this study introduces the HGWCA approach, which combines Grey Wolf Optimization with Cat Swarm Optimization. However, it faces two significant challenges. First off, there is not enough testing to determine how well the algorithm performs in actual cloud data centers. Second, due of its unclear scalability in larger data centers with numerous virtual machines, its effectiveness in managing extensive cloud settings is questioned. To confirm the algorithm's scalability and feasibility, these issues must be fixed. Future studies on improving makespan and throughput in cloud data centers can focus on several key areas by utilizing GWO and CSO algorithms. Among these are the following: enhancing algorithmic techniques through machine learning and hybridization; enabling dynamic resolution of scalability concerns in large data centers; exploring multi-objective optimization to consider additional performance objectives; managing real-time optimization to deal with evolving environments; and conducting practical implementation and validation studies to assess practical appropriateness. By taking these routes, scholars can contribute to the development of more reliable and effective cloud computing infrastructures that satisfy the evolving needs of cloud data center providers and users.

References

[1] Li, P., Li, J., Huang, Z., Li, T., Gao, C. Z., Yiu, S. M., & Chen, K. (2017). Multi-key privacy-preserving deep learning in cloud computing. Future Generation Computer Systems, 74, 76-85. https://doi.org/10.1016/j.future.2017.02.006

[2] Jivanadham, L. B., Islam, A. M., Katayama, Y., Komaki, S., & Baharun, S. (2013, May). Cloud Cognitive Authenticator (CCA): A public cloud computing authentication mechanism. In 2013 International Conference on Informatics, Electronics and Vision (ICIEV) (pp. 1-6). IEEE. https://doi.org/10.1109/iciev.2013.6572626

[3] Al-Maytami, B. A., Fan, P., Hussain, A., Baker, T., & Liatsis, P. (2019). A task scheduling algorithm with improved makespan based on prediction of tasks computation time algorithm for cloud computing. IEEE Access, 7, 160916-160926. https://doi.org/10.1109/access.2019.2948704

[4] Markovic, D. S., Zivkovic, D., Branovic, I., Popovic, R., & Cvetkovic, D. (2013). Smart power grid and cloud computing. Renewable and Sustainable Energy Reviews, 24, 566-577. https://doi.org/10.1016/j.rser.2013.03.068

[5] 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 & Information Systems, 14(7). https://doi.org/10.3837/tiis.2020.07.005

[6] Ujjwal, K. C., Garg, S., Hilton, J., Aryal, J., & Forbes-Smith, N. (2019). Cloud Computing in natural hazard modeling systems: Current research trends and future directions. International Journal of Disaster Risk Reduction, 38, 101188. https://doi.org/10.1088/1475-7516/2020/07/005

[7] Takahashi, T., Blanc, G., Kadobayashi, Y., Fall, D., Hazeyama, H., & Matsuo, S. I. (2012, April). Enabling secure multitenancy in cloud computing: Challenges and approaches. In 2012 2nd Baltic Congress on Future Internet Communications (pp. 72-79). IEEE. https://doi.org/10.1109/bcfic.2012.6217983

[8] Yang, C., Goodchild, M., Huang, Q., Nebert, D., Raskin, R., Xu, Y., ... & Fay, D. (2011). Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing? International Journal of Digital Earth, 4(4), 305-329. https://doi.org/10.1080/17538947.2011.587547

[9] Abdel-Basset, M., El-Shahat, D., El-Henawy, I., De Albuquerque, V. H. C., & Mirjalili, S. (2020). A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection. Expert Systems with Applications, 139, 112824. https://doi.org/10.1016/j.eswa.2019.112824

[10] Zamfirache, I. A., Precup, R. E., Roman, R. C., & Petriu, E. M. (2022). Policy iteration reinforcement learning-based control using a grey wolf optimizer algorithm. Information Sciences, 585, 162-175. https://doi.org/10.1016/j.ins.2021.11.051

[11] Rodríguez, L., Castillo, O., Soria, J., Melin, P., Valdez, F., Gonzalez, C. I., ... & Soto, J. (2017). A fuzzy hierarchical operator in the grey wolf optimizer algorithm. Applied Soft Computing, 57, 315-328. Faris, H., Aljarah, I., Al-Betar, M. A., & Mirjalili, S. (2018). Grey wolf optimizer: a review of recent variants and applications. Neural computing and applications, 30, 413-435. https://doi.org/10.1016/j.asoc.2017.03.048

[12] Bezdan, T., Zivkovic, M., Bacanin, N., Strumberger, I., Tuba, E., & Tuba, M. (2022). Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm. Journal of Intelligent & Fuzzy Systems, 42(1), 411-423. https://doi.org/10.3233/jifs-219200

[13] Sundas, A., Badotra, S., Alotaibi, Y., Alghamdi, S., & Khalaf, O. I. (2022). Modified Bat Algorithm for Optimal VM's in Cloud Computing. Computers, Materials & Continua, 72(2). https://doi.org/10.32604/cmc.2022.025658

[14] Raghavan, S., Sarwesh, P., Marimuthu, C., & Chandrasekaran, K. (2015, January). Bat algorithm for scheduling workflow applications in cloud. In 2015 International Conference on Electronic Design, Computer Networks & Automated

Verification (EDCAV) (pp. 139-144). IEEE. https://doi.org/10.1109/edcav.2015.7060555

[15] Fahim, Y., Rahhali, H., Hanine, M., Benlahmar, E. H., Labriji, E. H., Hanoune, M., & Eddaoui, A. (2018). Load Balancing in Cloud Computing Using Meta-Heuristic Algorithm. Journal of Information Processing Systems, 14(3). https://doi.org/10.1109/cist.2014.7016608

[16] Senthil Kumar, A. M., Padmanaban, K., Velmurugan, A. K., Asha Shiny, X. S., & Anguraj, D. K. (2023). A novel resource management framework in a cloud computing environment using hybrid cat swarm BAT (HCSBAT) algorithm. Distributed and Parallel Databases, 41(1-2), 53-63. https://doi.org/10.1007/s10619-021-07339-w

[17] Bin, N. I. N. G., Qiong, G. U., Zhao, W. U., Lei, Y. U. A. N., & Chun-yang, H. U. (2015). Bats algorithm research in cloud computing resource scheduling based on membrane computing. Application Research of Computers/Jisuanji Yingyong Yanjiu, 32(3). https://doi.org/10.4028/www.scientific.net/amr.989-994.2192

[18] Ganne, A. (2022). Emerging Business Trends in Cloud Computing. International Research Journal of Modernization in Engineering Technology, 4(12). https://doi.org/10.56726/irjmets32082

[19] Gundu, S. R., Panem, C. A., Thimmapuram, A., & Gad, R. S. (2022). Emerging computational challenges in cloud computing and RTEAH algorithm based solution. Journal of Ambient Intelligence and Humanized Computing, 1-15. https://doi.org/10.1007/s12652-021-03380-w

[20] Alam, T., Gupta, R., Qamar, S., & Ullah, A. (2022). Recent applications of Artificial Intelligence for Sustainable Development in smart cities. In Recent Innovations in Artificial Intelligence and Smart Applications (pp. 135-154). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-031-14748-7_8

[21] Ullah, A., & Chakir, A. (2022). Improvement for tasks allocation system in VM for cloud datacenter using modified bat algorithm. Multimedia Tools and Applications, 81(20), 29443-29457. https://doi.org/10.1007/s11042-022-12904-1

[22] Kumar, R., Bhardwaj, D., & Joshi, R. (2022). Adaptive bat optimization algorithm for efficient load balancing in cloud computing environment. In Advances in Computational Intelligence and Communication Technology: Proceedings of CICT 2021 (pp. 357-369). Singapore: Springer Singapore. https://doi.org/10.1007/978-981-16-9756-2_35

[23] Li, X., Lu, Y., Fu, X., & Qi, Y. (2021). Building the Internet of Things platform for smart maternal healthcare services with wearable devices and cloud computing. Future Generation Computer Systems, 118, 282-296. https://doi.org/10.1016/j.future.2021.01.016

[24] Krishnamoorthy, P. (2021). Performance Analysis of Hybrid BAT Algorithm and Cuckoo Search Algorithm [HB-CSA] for Task Scheduling in Mobile Cloud Computing. Available at SSRN 3997784. https://doi.org/10.2139/ssrn.3997784

[25] Gundu, S. R., Panem, C. A., & Thimmapuram, A. (2020). Hybrid IT and multi cloud an emerging trend and improved performance in cloud computing. SN Computer Science, 1(5), 256. https://doi.org/10.1007/s42979-020-00277-x

[26] Ullah, A., Nawi, N. M., & Khan, M. H. (2020). BAT algorithm used for load balancing purpose in cloud computing: an overview. International Journal of High-Performance Computing and Networking, 16(1), 43-54. https://doi.org/10.1504/ijhpcn.2020.110258

[27] Ibrahim, L. M., & Saleh, I. A. (2020). A solution of loading balance in cloud computing using optimization of bat swarm algorithm. Journal of Engineering Science and Technology, 15(3), 2062-2076. https://doi.org/10.5220/000767440058006

[28] Chung, K., & Park, R. C. (2019). Chatbot-based heathcare service with a knowledge base for cloud computing. Cluster Computing, 22, 1925-1937. https://doi.org/10.1007/s10586-018-2334-5

[29] Jian, C., Chen, J., Ping, J., & Zhang, M. (2019). An improved chaotic bat swarm scheduling learning model on edge computing. IEEE Access, 7, 58602-58610. https://doi.org/10.1109/access.2019.2914261

[30] Patil, R., Dudeja, H., & Modi, C. (2019). Designing an efficient security framework for detecting intrusions in virtual network of cloud computing. Computers & Security, 85, 402-422. https://doi.org/10.1016/j.cose.2019.05.016

[31] Lu, C., Gao, L., & Yi, J. (2018). Grey wolf optimizer with cellular topological structure. Expert Systems with Applications, 107, 89-114. https://doi.org/10.1016/j.eswa.2018.04.012

[32] Gawali, M. B., & Shinde, S. K. (2018). Task scheduling and resource allocation in cloud computing using a heuristic approach. Journal of Cloud Computing, 7(1), 1-16. https://doi.org/10.1186/s13677-018-0105-

[33] Attaran, M. (2017). Cloud computing technology: leveraging the power of the internet to improve business performance. Journal of International Technology and Information Management, 26(1), 112-137. https://doi.org/10.58729/1941-6679.1283

[34] Singh, P., Dutta, M., & Aggarwal, N. (2017). A review of task scheduling based on meta-heuristics approach in cloud computing. Knowledge and Information Systems, 52, 1-51. https://doi.org/10.1007/s10115-017-1044-2

[35] Zhang, Y., Liu, Z., Yu, F., & Jiang, T. (2017). Cloud computing resources scheduling optimisation based on improved bat algorithm via wavelet perturbations. International Journal of High-Performance Systems Architecture, 7(4), 189-196. https://doi.org/10.1504/ijhpsa.2017.092385

[36] Arunarani, A. R., Manjula, D., & Sugumaran, V. (2017). FFBAT: A security and cost-aware workflow scheduling approach combining firefly and bat algorithms. Concurrency and Computation:

Practice and Experience, 29(24), e4295. https://doi.org/10.1002/cpe.429

[37] Mittal, N., Singh, U., & Sohi, B. S. (2016). Modified grey wolf optimizer for global engineering optimization. Applied Computational Intelligence and Soft Computing, 2016. https://doi.org/10.1155/2016/7950348

[38] Bouyer, A., & Arasteh, B. (2014). The necessity of using cloud computing in educational system. Procedia-Social and Behavioral Sciences, 143, 581-585. https://doi.org/10.1016/j.sbspro.2014.07.440

[39] Dillon, T., Wu, C., & Chang, E. (2010, April). Cloud computing: issues and challenges. In 2010 24th IEEE international conference on advanced information networking and applications (pp. 27-33). Ieee. https://doi.org/10.1109/aina.2010.187

[40] 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. https://doi.org/10.1109/waina.2010.85

[41] Ullah, A., Razak, S. F. A., Yogarayan, S., & Sayeed, M. S. (2025). Modified Neural Network Used for Host Utilization Predication in Cloud Computing Environment. Computers, Materials & Continua, 82(3). https://doi.org/10.32604/cmc.2025.059355

[42] Remmach, H., Razak, S. F. A., Ullah, A., Yogarayan, S., Sayeed, M. S., & Mrhari, A. (2025). CNN-Based Multi-Output and Multi-Task Regression for Supershape Reconstruction from 3D Point Clouds. Informatica, 49(5). https://doi.org/10.31449/inf.v49i5.6863

[43] Ullah, A., Alam, T., Aziza, C., Sebai, D., & Abualigah, L. (2024). A Hybrid Strategy for Reduction in Time Consumption for Cloud Datacenter Using HMBC Algorithm. Wireless Personal Communications, 137(4), 2037-2060. https://doi.org/10.1007/s11277-024-11395-7

[44] Alam, T., Gupta, R., Nasurudeen Ahamed, N., Ullah, A., & Almaghthwi, A. (2024). Smart mobility adoption in sustainable smart cities to establish a growing ecosystem: Challenges and opportunities. MRS Energy & Sustainability, 11(2), 304-316. https://doi.org/10.1557/s43581-024-00092-4

[45] Ullah, A., Alomari, Z., Alkhushayni, S., Al-Zaleq, D. A., Bany Taha, M., & Remmach, H. (2024). Improvement in task allocation for VM and reduction of Makespan in IaaS model for cloud computing. Cluster Computing, 27(8), 11407-11426. https://doi.org/10.1007/s10586-024-04539-8

[46] Alam, T., Ullah, A., & Benaida, M. (2023). Deep reinforcement learning approach for computation offloading in blockchain-enabled communications systems. Journal of Ambient Intelligence and Humanized Computing, 14(8), 9959-9972. https://doi.org/10.1007/s12652-021-03663-2

[47] Ouhame, S., Hadi, Y., & Ullah, A. (2021). An efficient forecasting approach for resource utilization in cloud data center using CNN-LSTM model. Neural Computing and Applications, 33(16), 10043-10055. https://doi.org/10.1007/s00521-021-05770-9

[48] Clemons, E. K., & Chen, Y. (2011, January). Making the decision to contract for cloud services: Managing the risk of an extreme form of IT outsourcing. In 2011 44th Hawaii International Conference on System Sciences (pp. 1-10). IEEE. https://doi.org/10.1109/hicss.2011.292

[49] Yao, J., & He, J. H. (2012, April). Load balancing strategy of cloud computing based on artificial bee algorithm. In 2012 8th International conference on computing technology and information management (NCM and ICNIT) (Vol. 1, pp. 185-189). IEEE. https://doi.org/10.3724/sp.j.1087.2012.0244

[50] Sajid, M., & Raza, Z. (2013, December). Cloud computing: Issues & challenges. In International conference on cloud, big data and trust (Vol. 20, No. 13, pp. 13-15). sn. https://doi.org/10.1201/b16318-3

[51] Pawar, C. S., & Wagh, R. B. (2013, March). Priority based dynamic resource allocation in cloud computing with modified waiting queue. In 2013 International Conference on Intelligent Systems and Signal Processing (ISSP) (pp. 311-316). IEEE. https://doi.org/10.1109/issp.2013.6526925

[52] Chauhan, R., & Kumar, A. (2013, November). Cloud computing for improved healthcare: Techniques, potential and challenges. In 2013 E-health and bioengineering conference (EHB) (pp. 1-4). IEEE. https://doi.org/10.1109/ehb.2013.6707234

[53] Velugoti, S., & Vani, M. P. (2024). An approach for privacy preservation assisted secure cloud computation. Informatica, 47(10). https://doi.org/10.31449/inf.v47i10.4586

[54] Zhang, R., & Shi, W. (2020). Research on resource allocation and management of mobile edge computing network. Informatica, 44(2). : https://doi.org/10.31449/inf.v44i2.3166

[55] Debbi, H. (2021). Modeling and performance analysis of resource provisioning in cloud computing using probabilistic model checking. Informatica, 45(4). DOI: https://doi.org/10.31449/inf.v45i4.3308