Efficient task scheduling in the cloud with queuing and multi-tactic harris hawks optimization

Sheetal Antony[1*], Sujatha S R[2]

[1]Ajosha Bio Teknik Pvt. Ltd

Computer Science and Engineering, Sri Siddhartha Institute of Technology, SSAHE

SSIT Maralur, Tumakuru, 572105.

e-mail: sheetal123876@gmail.com

(Received 24 February 2024; Final version received 18 July 2024; Accepted 31 July 2024)

Abstract

Cloud computing faces challenges in task scheduling, which is crucial for cost-efficient execution and resource utilization. Current methods face computational complexity, especially in large-scale data centres. This paper proposes a novel approach that considers job dependencies and task execution times to reduce make-span, minimize energy consumption, and balance resource loads. VMs are allocated based on workflow task requirements, using thresholds for task levels and durations to manage execution priorities. Tasks with higher dependencies and longer execution times are prioritized, ensuring efficient resource utilization and energy savings. The method employs queues for different task intensities, streamlining VM allocation by organizing tasks with additional metadata like intensities, arrival times, and deadlines. Historical scheduling logs (HSLs) are used to generate appropriate VMs, with new VMs created if no matching records exist in the HSLs. The proposed solution optimizes scheduling using an enhanced Multi-Tactic Harris Hawks Optimization (MTHHO) algorithm, which addresses the limitations of traditional HHO by incorporating Sobol sequences, elite opposition-based learning, and improved energy updating techniques to enhance population diversity, adaptability, and convergence accuracy while avoiding local optima using the Gaussian walk learning. The result shows that the proposed method of QoS performances attained less Makespan, energy consumption of 0.20, throughput of 2.4, and execution time of 16.75 with effectively allocated resources of 98% when compared to the previous methods in cloud computing. Therefore, the proposed heuristic-based MTHHO method balanced the load and allocated the resources effectively to improve QoS performances.

*Keywords: Cloud Computing; Thresholds; Energy Consumption; Queuing; Task Scheduling; Multi-Tactic HHO; non-linear weight; Gaussian walk learning ; Load Balancing; Makespa*n*;*

1. Introduction

Cloud computing systems were created based on the enormous growth in internet data processing. When it comes to giving technology facilities online, cloud computing is crucial. Without direct active control, it gives users access to computer system resources like data storage and processing power. Three different services about infrastructure, platforms, and software can be offered by a cloud. Infrastructure as a Service (IaaS), which offers infrastructure services including storage systems and computing resources, is the first service. Platform as a service (PaaS), the second offering, allows customers to generate presentations based on the platform that is made available. The third service, known as software as a service (SaaS), offers customers the option of using software straight from the cloud without having to install anything locally [Devaraj etc., 2020, Alam 2021]. Cloud computing allows for the flexible and elastic provision of varied computing resources in response to user demands; in recent years, such large-scale applications have used cloud computing at an increasing rate [Cui etc., 2021]. A cloud data centre's infrastructure typically comprises thousands of big computing hosts with fast computing power [Katal etc., 2023].

Virtual machines (VMs) are a type of computing resource that cloud providers utilize to deliver computing resources to users. To increase the general effectiveness of cloud computing, effective task scheduling is needed when several users make task requests for services from the cloud. Efficient task scheduling allows the optimal resource allocation between the requested tasks over a limited amount of period enabling the achievement of the desired degree of quality of service (QoS) [Adhikari etc., 2019, Gawalil etc., 2018]. The primary categories of scheduling mechanisms used in cloud computing are workflow, static, cloud service, and dynamic scheduling [Alsaidy etc., 2022]. The difficulty of the process allows for the classification of scheduling methods as heuristic, meta-heuristic, and hybrid task scheduling techniques. Static scheduling employs heuristic approaches such as minimum execution time (MET), minimum completion time (MCT), shortest job to fastest processor (SJFP), longest job to fastest processor (LJFP), Min-Min, and Max-Min [Abd etc., 2019, Mishra etc., 2020]. In a cloud environment, it can be challenging to schedule tasks and allocate resources in the best possible order and with the least amount of delay to increase system presentation. The complexities of the cloud, real-time task mapping to virtual machines, and virtual machine mapping to the host machine make task scheduling in cloud computing an NP-Hard problem [Yadav etc., 2023, Golchi etc., 2019].

Meta-heuristic algorithms have recently captured the interest of researchers due to their capacity to solve large-scale issues efficiently. For NP-Hard problems, these algorithms can efficiently search a wide area of the solution space for a solution that is close to optimal [Abdullahi etc., 2023]. Meta-heuristic algorithms like the genetic algorithm (GA) [Keshanchi et at., 2017], ant colony optimization (ACO) [Mahato etc., 2017], particle swarm optimization (PSO) [Mansouri etc., 2019, Kumar etc., 2018], discrete symbiotic organism search (DSOS) [Abdullahi etc., 2016], and gravitational search algorithm (GSA) [Chaudhary etc., 2018] have recently been used to solve task scheduling issues [Agarwal etc., 2021, Wei 2020]. However, metaheuristic algorithms possess two major shortcomings: The first is that they are computationally intensive and can get stuck in local optimum states, particularly in large solution spaces. According to Konjaang and Xu Ramamoorthy et al [Konjaang etc., 2021, Ramamoorthy etc., 2021], an inequity among local and global search strategies may origin convergence to occur too early.

Hybrid meta-heuristic algorithms were employed by the researchers to achieve improved performance. Examples of this hybridization include the hybridized whale optimization method [Strumberger etc., 2019], firefly and PSO [1, 11], Q-learning and PSO [Jena etc., 2022], as well as firefly and simulated annealing [Fanian etc., 2018]. The contribution of the work is outlined as:

- Energy-efficient VMs are crucial for cost-effective, elastic computing in cloud data centres, but their computational complexity can limit their usability in dynamic environments and hinder real-time responsiveness and scalability.
- A proposed approach focuses on job dependencies and task execution times to shorten makespan, consume less energy, and balance the load on available resources.
- Tasks are assigned VMs based on workflow tasks, with tasks with longer execution times handled first. Queues are maintained based on job intensities, and tasks are stored in queues using additional information.
- The enhanced Multi-Tactic Harris Hawks Optimization (MTHHO) algorithm is used to optimize scheduling issues. The algorithm uses Sobol sequences, elite opposition-based learning, and the Gaussian walk learning technique to improve the population's variety, adaptability, and energy updating.

The rest of the manuscript is organized as given. An introduction is given in Section 1, related works are presented in Section 2, and effective task scheduling in cloud computing using queuing and multi-tactic Harris Hawks Optimisation modelling is covered in Section 3. Sections 4 as well as 5 provide the model results, and the conclusion correspondingly.

2. Related Works

Several studies are involved on the topic of cloud computing, including scheduling, load balancing, and resource provisioning. Numerous problems with cloud computing have captured the attention and concern of researchers. Resource management, load balancing, cloud migration, privacy and security, energy consumption, availability and scalability, interoperability, and compatibility are a few of these significant concerns. These challenges are strongly influenced by investigating effective task scheduling. The fundamental to task scheduling in cloud computing is to identify the most optimal mapping connection among tasks and virtual machines depending on the objectives of users and cloud systems. One of the main ways to deal with this issue is to find a more effective algorithm to support job scheduling, such as a single-objective optimization algorithm or a multi-objective optimization algorithm. The following section discusses some of the utmost recent task-scheduling approaches.

Kaur, etc., outlined the major research requirements for load balancing optimization in the prior works that must be filled to address the load balancing problem in cloud environments. To maximize the use of VMs with uniform load distribution, a framework for resource provisioning and a combination method for load balancing has been created in the current work. The suggested system is based on the fusion of heuristic techniques with meta-heuristic algorithms to achieve the greatest efficiency in making span and cost. For the HDD-PLB system, two hybrid methods have been proposed: the Hybrid Heterogeneous Earliest Finish Time (HEFT) Heuristic with ACO (HHA) and the Hybrid Predict Earliest Finish Time (PEFT) Heuristic with ACO meta-heuristic (HPA). The two load-balancing approaches have been analysed and contrasted for the suggested HDD-PLB system to determine which is superior. However, the suggested framework is based on financial restraints, limiting the execution of workflow tasks that exceed deadlines in terms of total cost. The normal time and cost findings of the 100 repetitions were not measured for this research.

Kruekaew, etc., suggested the MOABCQ method as a standalone task scheduling method for cloud computing to tackle workload balancing problems with a Multi-objective task scheduling optimization depending on the Artificial Bee Colony Algorithm (ABC) with a Q-learning algorithm, which is a reinforcement learning method that assists the ABC process work more rapidly. The proposed solution addresses the limitations of simultaneous concerns by maximizing VM throughput, optimizing scheduling and resourceconsumpti on, and establishing load balancing among VMs according to make span, cost, and resource utilization. The efficiency study of the suggested approach was contrasted utilizing CloudSim with the load balancing and scheduling methods currently in use: Max-Min, FCFS, HABC_LJF, Q-learning, MOPSO, and MOCS in three datasets: Random, Google Cloud Jobs (GoCJ), and Synthetic workload. According to the findings, MOABCQ-based algorithms beat other algorithms on account of lowering makespan, cost, degree of imbalance, boosting throughput, and utilizing resources regularly. However, it cannot ensure that the MOABCQ_LJF method is best, and also not all test datasets can be used to optimize the system's performance.

Velliangiri etc. proposed a Hybrid Electro Search with a Genetic Algorithm (HESGA) to improve work scheduling performance by accounting for aspects such as makespan, balance of load, utilisation of resources, and multi-cloud costs. The proposed method integrates the advantages of genetic and electro-search algorithms. The Electro search method generates the finest global optimum results, while the GA generates the finest local optimal results. The suggested technique outperforms current scheduling methods as the Hybrid Particle Swarm Optimization Genetic technique (HPSOGA), GA, ES, and ACO. However, there are no guarantees that the algorithm will find the globally best solution or that it will always lead to optimal resource use, which is essential for cloud computing to be cost-effective.

Rajakumari, etc., proposed the Fuzzy Based Ant Colony Optimization Scheduling approach, however, is used in cloud computing to address task scheduling issues like optimal task scheduling presentation results. First, by suggesting a Dynamic Weighted Round-Robin method, work scheduling performance in the cloud is enhanced. The performance of work scheduling is enhanced by the suggested DWRR algorithm by taking into account resource competence, task priority, and length. Next, a hybrid particle swarm parallel ant colony optimization heuristic approach is suggested to address the task execution delay issue in DWRR-based task scheduling. Finally, HPSPACO develops a fuzzy logic system that enhances job scheduling in the cloud system. For the inertia weight updates of the PSO and pheromone trails updating of the PACO, a fuzzy technique is suggested. To optimize work scheduling, the proposed Fuzzy HPSPACO on cloud computing reduces implementation and waiting times boosts system throughput and maximizes resource usage. Conversely, the CF-ACO algorithm is a complex algorithm with multiple elements, including fuzzy logic and ant colony optimization, which can result in fundamental processing overhead and difficulties in finding optimal solutions in real-time or large-scale cloud systems.

Saxena, D., etc., proposed a unique secure and multi-objective virtual machine placement (SM-VMP) model using a successful VM migration to tackle the problems of resource waste, excessive usage of power, higher inter-communication costs, and security breaches. The suggested approach emphasizes the safe and quick operation of user applications by minimizing inter-communication latency and enabling an energy-efficient allocation of physical resources across VMs. The VMP is implemented using the suggested Whale Optimisation Genetic Algorithm (WOGA), which is motivated by non-dominated sorting-based genetic algorithms and whale evolutionary optimization. The results of the assessment for static and dynamic VMP as well as contrasting it using contemporary advanced revealed a considerable drop in shared servers, intercommunication costs, power consumption, and processing period up to 28.81%, 25.7%, 35.9%, and 82.21%, correspondingly. Resource utilization also improved up to 30.21%. However, multi-objective optimization issues in cloud data centres can be computationally complex, time-intensive, and difficult to solve.

3. Proposed Methodology

The allocation of energy-efficient virtual machines (VMs) is crucial for cost-effective and elastic computing in cloud data centres. However, the computational complexity of existing work, particularly in large-scale data centres and complex optimization objectives, can limit the framework's usability in dynamic, resource-intensive environments and impair its real-time responsiveness and scalability. Multi-objective optimization can be computationally intensive and time-consuming. It is crucial to understand how computational complexity impacts the usability of the systems. High computational complexity can cause delays in VM allocation, impacting cloud service responsiveness and performance. As data centres scale, managing resource complexity becomes more complex, requiring clear management strategies to ensure system effectiveness at larger scales. Complex algorithms may require more computational power, potentially offsetting energy savings achieved through efficient VM allocation. Implementing sophisticated algorithms can pose challenges, but clear guidelines and understanding can help ensure smooth deployment and operation. The suggested approach focuses on job dependencies and task execution times to shorten make-span, consume less energy, and balance the load on available resources. To preserve a balanced load on the resources, VMs are assigned tasks depending on the necessities of the workflow tasks. The suggested method scans the workflow tasks and establishes thresholds for the tasks' level and duration. When tasks are being executed, the threshold values are employed to handle them based on various priorities.

Tasks with higher dependencies generate system bottlenecks and extended execution durations. To shorten execution time, tasks with longer execution times are handled first, requiring high-priority processing and allocating VMs with powerful processing capabilities. The algorithm also prioritizes tasks with lengthy execution times, setting thresholds for dependencies and duration. This shortens execution time and reduces energy consumption by effectively utilizing resources. The suggested approach then employs queues based on the job intensities, maintaining distinct queues for CPU-intensive tasks and tasks with higher dependents. It takes less time to discover the right VMs during the VM allocation process when tasks of different intensities are placed in distinct queues. Tasks are stored in queues using additional information about them, such as their intensities, arrival times, and deadlines. The following step is to generate appropriate VMs for the tasks following the classification of tasks into various queues. Historical scheduling Logs HSLs are employed for this objective. The HSLs are updated appropriately and a novel VM is generated with the resources required to complete the task if there is no matching record in the HSLs. The suggested algorithm optimizes the scheduling issue using the enhanced Multi-Tactic Harris Hawks Optimization (MTHHO) algorithm. Scheduling with MTHHO begins following the pre-processing step. An enhanced MTHHO algorithm is suggested to make up for the traditional HHO process's low convergence accuracy, slower degree of convergence, and easy tendency to prey to the illusion of local optima. To improve the population's variety, Sobol sequences are first utilized to start the population. The next step to raise the adaptability and the standard of the solution sets, the elite opposition-based learning technique is used. Additionally, the original algorithm's energy updating technique has been improved to increase the process's capability to explore as well as exploit in a nonlinear update way. To prevent the process from being stuck and settling into a local optimum, the Gaussian walk learning technique is finally employed. The suggested framework for task scheduling utilizing the enhanced MTHHO process is shown in Fig 1.

The proposed approach to task scheduling optimizes resource utilization, reduces make-span, and saves energy by using energy-efficient virtual machines (VMs) and optimizing scheduling. It considers job dependencies and task execution times, maximizing resource allocation, reducing idle times, and minimizing make-span. The approach prioritizes tasks based on dependencies and execution times, ensuring critical tasks are completed promptly. It also balances resource loads across the data centre, preventing overutilization or underutilization, and improving the stability and reliability of the cloud infrastructure. The enhanced MTHHO algorithm, using advanced techniques like Sobol sequences and elite opposition-based learning, enhances scheduling accuracy, leading to better convergence on optimal solutions and avoiding common pitfalls like local optima. Overall, this approach significantly improves task scheduling efficiency and reduces energy consumption.

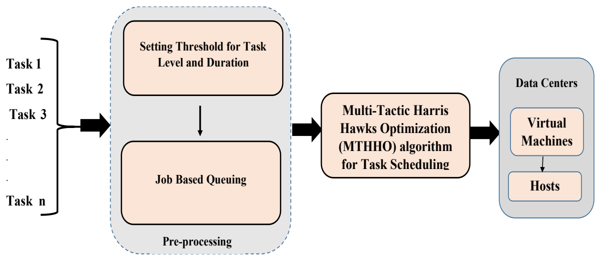
Task 1
Task 2
Task 3
Task n
Setting Threshold for Task Level and Duration
Job Based Queuing
Pre-processing
Multi-Tactic Harris Hawks Optimization (MTHHO) algorithm for Task Scheduling
Data Centers
Virtual Machines
Hosts

Fig 1. The Proposed Framework for Task Scheduling

3.1 Materials and Approaches

The proposed method is explained further in this part of the article. The process focuses on load balance, makespan, and energy consumption. The process is separated into dual phases: MTHHO-based optimization and preprocessing. The process and cloud framework are covered initially, trailed by the information of every stage.

3.2 Workflow and Cloud Architecture

Workflow programmes are made up of tasks that have reliance, such as implementation and data reliance. The tasks in the previous examples have a parent-child relationship. When the execution of each parent's job has finished, the child's task can begin. In the latter scenario, the tasks exchange data, meaning that the result produced by one job is used as the input for another. It is challenging for a scheduler to efficiently schedule resources for workflow programs because of these dependencies. A directed acyclic graph (DAG), such as $D\ (V, E)$, is used to represent workflow activities. In this graph, E stands for the edges while V stands for the vertices. VMs, or virtualized resources, are the building blocks of cloud computing. A scheduling process's objective is to assign R_i to W_j, where W_j is the workflow application $(W_1, W_2, W_3, \dots, W_m$ and R_i is the ith resource from a pool of VMs $(VM_1, VM_2, VM_3, \dots, VM_n)$. The objective is to reduce energy usage and implementation period while maintaining a balanced load on the available resources. Processing, memory, storage, bandwidth, and other capacities have been pre-allocated for the resources. Equation (1) illustrates how the number of processing elements (PEs) and MIPs of each PE are used to calculate the processing capacity (C_i) of a resource VM_i.

$$C_i = (PE_{\times} MIPS_i) \tag{1}$$

Equation (2) is used to determine the capacity of n resources, or virtual machines.

$$C = \sum_{i=1}^{n} C_i \tag{2}$$

Every VM has a resource utilisation at any given period, which is called the VM load. Equation (3) is used to compute the load, where TL the total length of tasks that VM_i is processing and Ci is VM_i's capacity.

$$L_{vmi} = \frac{TL}{C_i} \tag{3}$$

Equation (4) is utilised to determine each VM's load L.

$$L = \sum_{i=1}^{n} L_{vmi} \tag{4}$$

Equation (5) illustrates how load balancing is calculated as the load across various cloud environment nodes.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (L_{vmi} - \bar{L})^2}{n}} \tag{5}$$

where $\bar{L}$ is the average load across all VMs, n is the amount of VMs, and L_{vmi} is the load of VM_i. The way resources are used has a big impact on how much energy cloud computing uses. Equation (6) can be used to compute the utilisation.

$$U = \alpha \frac{\sum_{i=1}^{n} c_i}{C} + \beta \frac{\sum_{i=1}^{n} m_i}{M} \tag{6}$$

where n is the number of VMs running on host h, and c_i, m_i denotes the computing and memory assigned to VM_i. In Eq. (6), C and M are the total processing ability and memory of the host, and α and β are the weight factors of every resource.

Equation (7) can be used to compute the energy consumption, with $\boldsymbol{k}$ standing for the functioning energy consumption, or idle mode. $\boldsymbol{U}$ is the host resource utilisation determined by Equation (6), and $\boldsymbol{E_{max}}$ denotes the energy consumption during the processors' peak utilisation.

$$E_c = E_{idle} + (E_{max} - E_{idle}) \times U \quad (7)$$

Data processing may be required for workflow tasks. The workflow's total completion time accounts for both processing time and time spent acquiring the necessary data. The completion time of task t_i is calculated using Equation (8).

$$Time(t_i) = Time\left(\left(Trans_{t_i}, t_j\right) + Time_E(t_i, VM_k)\right) \quad (8)$$

The time required to transmit data from task t_i to task t_j is denoted by $\left(Trans_{t_i}, t_j\right)$ in Equation (8), while the execution time of t_i on VM_k is represented by $Time_E(t_i, VM_k)$. Equations (9) and (10) are used to determine the two parameters, respectively.

$$Trans\left(t_i, t_j\right) = \frac{sizeof(t_i, t_j)}{\beta(VM_k, VM_m)} \quad (9)$$

The quantity of data that task t_i transfers to task t_j is represented by $sizeof\left(t_i, t_j\right)$ in Equation (9), and the bandwidth consumed by VM_k and VM_m is represented by $\beta(VM_k, VM_m)$. The cost of transmission is disregarded if both virtual machines are placed in the same data centre.

$$T_E = \frac{l_i}{C_{m_j}} \quad (10)$$

where C_{m_j} is the processing capability ofVM_j, which was determined using Equation (1), and l_i is the length of job i. A workflow's makespan is the entire period it gains to complete every task. Equation (11), where MS is the makespan and FT is the task's completing time, can be used to compute the makespan.

$$MS = FT_{i=1}^{n}[task_i time] \quad (11)$$

Energy-efficient virtual machines (VMs) significantly reduce operational costs by reducing power consumption in data centres, a crucial factor in a competitive market. They also support the elastic nature of cloud computing, allowing data centres to dynamically scale resources based on demand. This elasticity ensures optimal resource utilization without unnecessary energy expenditure. Additionally, the proposed method simplifies the allocation process by considering job dependencies and task execution times, making it more manageable and efficient even in complex scenarios.

3.3 Harris Hawks Optimization (HHO)

The HHO process is a mathematical explanation of the Harris hawk's technique for catching prey under various circumstances. Each iteration's best answer is regarded as the prey, while individual Harris hawks form candidate solutions. The process is allocated into two primary phases: exploration and exploitation. The amount of the prey's escape energy determines when to switch between the two phases. Below is a description of the original Harris Hawks optimisation algorithm.

3.3.1 Exploration Phase

The position data of the Harris hawk population largely determines the global search phase, and its update methodology is as follows:

$$Z(t+1) = \begin{cases} Z_{rand}(t) - r_1 |Z_{rand}(t) - 2r_2 Z(t)| & q \geq 0.5 \\ (Z_{prey}(t) - Z_m(t)) - r_3(LB + r_4(UB - LB)) & q < 0.5 \end{cases} \quad (12)$$

In this case, $r_1 - r_4$ and q are casually created among (0,1), being transformed every iteration; $Z(t+1)$ stands for the position of the hawks in the iteration $t+1$; $Z_{prey}(t)$ signifies the location of the prey; $Z(t)$ signifies the position of the hawks in the present generation t; UB and LB are the upper and lower bounds of the population, accordingly; $Z_{rand}(t)$ indicates a casually nominated hawk from the present population; and $Z_m(t)$ stands for the mean of individuals in the current population, which comes from Eq.(13):

$$Z_m(t) = \frac{1}{n}\sum_{k=1}^{n} Z_k(t) \quad (13)$$

where $Z_k(t)$ represents the location of hawk k in the reiteration t as well as n represents the number of hawks.

3.3.2 Conversion from Exploration to Exploitation

The following is the energy calculation that regulates the prey's outflow:

$$E_f = 2E_{esc0}(1 - t/T) \tag{14}$$

Where T is the maximum amount of repetitions, t is the number of repetitions that are currently in progress, and the rate of E_{esc0} is an arbitrary number between -1 and 1 that represents the energy's starting state. The global exploration phase is represented by the Harris hawks searching for the prey in various places while the escape energy $|E_{esc}| \geq 1$, and the local exploitation phase is represented by the Harris hawks searching the nearby solutions when $|E_{esc}| < 1$.

3.3.3 Exploitation Phase

Based on the findings of the previous stages' exploration, the Harris hawk will besiege the intended prey in this phase while it attempts to get away from the chase. For this stage of the simulation, four potential ways are suggested based on the actions of the Harris's hawk and its prey. E_{esc} simulates both the hard and soft besiege of Harris's hawk. The parameter r indicates whether or not the prey successfully escapes.

3.3.4 Soft Besiege

The prey attempts to get away from the hunt when $|E_{esc}| \geq 0.5$ and $r \geq 0.5$, at which point the Harris hawk uses a soft besiege to slowly deplete the prey's energy. The following is a model of the behaviour:

$$Z(t+1) = \Delta Z(t) - E_{esc}|IZ_{prey}(t) - Z(t)| \tag{15}$$

$$\Delta Z(t) = Z_{prey}(t) - Z(t) \tag{16}$$

where I is added to resemble the movement of the prey and r_5 is a randomly generated number between 0 and 1, with a random variation in its value for each iteration.

3.3.5 Hard Besiege

The Harris hawk uses Equation (17) to update its current position and launches a hard besiege attack on prey when it has insufficient energy to get away, specifically when $|E_{esc}| < 0.5$ and $r > 0.5$.

$$Z(t+1) = Z_{prey}(t) - E_{esc}|\Delta Z(t)| \tag{17}$$

3.3.6 Soft Besiege Using Increasingly Fast Dives

The prey has sufficient energy to outflow the hunt when $|E_{esc}| > 0.5$ and $r < 0.5$. Harris hawks will modify their positions by Equation (18):

$$X = Z_{prey}(t) - E_{esc}|IZ_{prey}(t) - Z(t)| \tag{18}$$

$$Y = X + S \times LF(D) \tag{19}$$

where D is the problem dimension, S is a random vector of size $1 \times D$, and LF is the levy flight function, which can be defined as in Eqn (20):

$$\begin{cases} LF(z) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}} \\ \sigma = \left(\frac{\Gamma(1+\beta) \times sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{\frac{1}{\beta}} \end{cases} \tag{20}$$

where β is the default constant, set at 1.5, and u and v are random numbers within the interval $(0,1)$. Therefore, Equation (21) can be used to carry out the last plan for apprising the Hawks' positions over the soft siege phase:

$$Z(t+1) = \begin{cases} X, if F(X) < F(Z(t)) \\ Y, if F(Y) < F(Z(t)) \end{cases} \tag{21}$$

where X and Y are gained using Equations (18) and (19), correspondingly.

3.3.7 Hard Besiege Using Increasingly Fast Dives

The prey lacks the energy to accomplish an outflow when $|E_{esc}| < 0.5$ and $r < 0.5$. In these cases, the following tactic is intended to be used:

$$Z(t+1) = \begin{cases} X, if F(X) < F(Z(t)) \\ Y, if F(Y) < F(Z(t)) \end{cases} \quad (22)$$

$$X = Z_{prey}(t) - E_{esc}|IZ_{prey}(t) - Z_m(t)| \quad (23)$$

$$Y = X + S \times LF(D) \quad (24)$$

where $Z_m(t)$ is gained using Equation (13)

3.3.8 Main Steps of HHO

Algorithm 1 depicts the important phases of the overall HHO algorithm.

Algorithm 1 Principal phases of the HHO process

```
Input: Population size $N$ and the maximum number of iterations $T$
1: Start the population
2: while $t < T$ do
3: Compute the suitability of every solution and get the best individual
4: for $i = 1: N$ do
5: Modify the escape energy $E_{esc}$ by Eq. (14)
6: if $|E_{esc}| \geq 1$ then
7: Modify the position by Eq. (12)
8: else if then $|E_{esc}| < 1$
9: if $|E_{esc}| \geq 0.5$ and $r \geq 0.5$ then
10: Modify the position by Eq. (15)
11: else if then $|E_{esc}| < 0.5$ and $r \geq 0.5$
12: Modify the position by Eq. (17)
13: else if then $|E_{esc}| \geq 0.5$ and $r < 0.5$
14: Modify the position by Eq. (21)
15: else if then$|E_{esc}| < 0.5$ and $r < 0.5$
16: Modify the position by Eq. (22)
17: end if
18: end if
19: end for
20: $t = t + 1$
21: end while
22: return $Z_{prey}$
```

3.4 Enhanced Multi-Tactic Harris Hawks Optimization (MTHHO)

The HHO process is effective for local development since it includes several development modes and alternates between them, but it also has the drawback of being vulnerable to the local optimum issue. To address this shortcoming, four enhancement techniques are employed in this article that enhance the initial algorithm. The MTHHO algorithm is a new approach to task scheduling optimization that overcomes the limitations of traditional HHO. It uses sobol sequences to initialize the population, ensuring uniform and comprehensive coverage of the search space. This diversity is crucial for exploring a wide range of potential solutions and avoiding premature convergence. Elite opposition-based learning improves the algorithm's ability to escape local optima by considering both current best solutions and their opposites, maintaining a balance between exploration and exploitation. Additionally, the algorithm includes refined energy updating mechanisms that better simulate the energy dynamics of hawks in nature, enhancing convergence accuracy and efficiency. This algorithm also employs Gaussian Walk Learning (GWL) to enhance population diversity and avoid local optima by initially using larger disruptions that rapidly decrease in later stages, thereby balancing algorithm creation and searchability.

3.4.1 Populations for Sobol Sequence Initialization

The speed and precision of the intelligent algorithm's convergence are significantly influenced by the primitive solution's distribution within the solution space. Randomization is used in the basic HHO method to create the initialised population. Nevertheless, the individuals produced in this manner are not uniformly dispersed around the exploration space, which consequently impacts the algorithm's accuracy and convergence rate. In contrast to the random sequence, the even distribution of points in space is a characteristic of the probabilistic low-difference Sobol sequence. The real population produced by the Sobol sequence can be given as follows:

$$Z_i = Lb + S_n \times (Ub - Lb) \quad (25)$$

where S_n is the random number produced by the Sobol sequence, where $S_n \in [0, 1]$, and Lb and Ub are the exploration space's lower and upper limits, accordingly.

The original population space distribution is compared between the random initialization and the Sobol sequence initialization population spaces in Fig 2, assuming that the population size is 100, the search space is two-dimensional, and the upper and lower limits are 1 and 0, correspondingly.

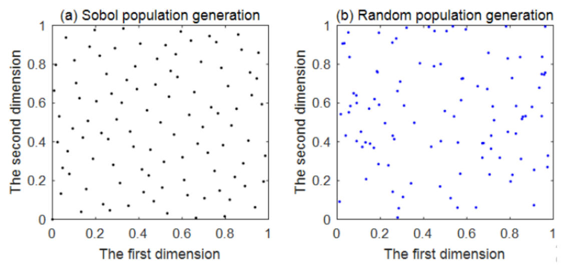

Fig 2. Comparison of Sobol and random population generation

The real population produced by the Sobol sequence is further evenly distributed, as seen in Fig 2, which allows the optimization process to conduct a superior global exploration in the exploration space. This increases the population's diversity and accelerates the algorithm's rate of convergence.

3.4.2 Elite Opposition-Based Learning

Opposition-based learning (OBL) is a successful method of intelligent computing developed by Tizhoosh in 2005. This technique has been used recently to enhance several algorithms and has shown excellent optimization outcomes. Considering an instance where a feasible response in d-dimensional search space is $Z = (z_1, z_2, \cdots, z_d)(z_j \in [a_j, b_j])$, then the definition of its opposition-based solution is $\bar{Z} = (\overline{z_1}, \overline{z_2}, \cdots, \overline{z_d})$, where $\overline{z_j} = r(a_j + b_j) - z_j$, r is the uniform distribution coefficient between $[0, 1]$.

The inverse solution developed by the opposition-based learning technique does not always search for the global optimal solution simpler than the existing exploration space. Elite opposition-based learning (EOBL) is suggested as a solution to this issue. Considering that the elite individual represents the extreme of the current population in the search space $Z_e = (z_1^e, z_2^e, \cdots, z_d^e)$, then its inverse solution $\overline{Z_e} = (\overline{z_1^e}, \overline{z_2^e}, \cdots, \overline{z_d^e})$ can be stated as follows:

$$\overline{z_j^e} = k \cdot (a_j + b_j) - z_j^e \qquad (26)$$

where $a_j = min(z_j^e), b_j = max(z_j^e), k$ is a random number inside $[0, 1]$, and $z_j^e \in [a_j, b_j]$. It also has a dynamic border with upper and lower limits, respectively, represented by b_j and a_j. It is advantageous for the produced inverse solution to slowly decrease the search space and accelerate the algorithm's convergence by substituting a dynamic boundary for the fixed boundary. The method used to reset the value is as follows to prevent the elite inverse solution from jumping beyond the boundary and losing its viability:

$$\overline{z_i^e} = rand(a_j, b_j) \qquad (27)$$

3.4.3 Optimisation of Escape Energy Update

The energy factor E_{esc} is used by a Harris hawk in the basic HHO to control the algorithm's shift from the global search stage to the local search stage. Nevertheless, as Eq. (14) illustrates, a linear update is used to lower its energy factor E_{esc} from $2\ to\ 1$, this, in the latter part of the cycle, results in locking it in a local optimum. When the process advances to the latter stage, a novel, upgraded form of the energy factor is utilized to address the drawback of just local searching:

$$E_{esc} = \begin{cases} \cos(\pi \times (t/T + 1/2) + 2), t \leq T/2 \\ \cos(\pi \times (t/T - 1/2)^{1/3}), t > T/2 \end{cases} \qquad (28)$$

$$E_{esc1} = E_{esc} \times (2 \times rand - 1) \qquad (29)$$

where r is the random number inside $[0, 1]$, T is the maximum number of iterations, and t is the number of iterations that are currently being done.

As shown in Fig 3, the algorithm's global search capability is controlled by a fast deceleration rate early in the iteration. The lowering rate reduces down in the middle of the iteration to equilibrium the capabilities of local exploitation and global exploration. Later in the iteration, the local search picks up speed and its value decreases quickly. Fig 4 shows that E_{esc1} has changing energy parameters all over the recursive procedure and has the capability of both global and local searching, with global exploration taking place primarily in the initial phase and more local exploitation taking place later while still maintaining the potential of global exploration.

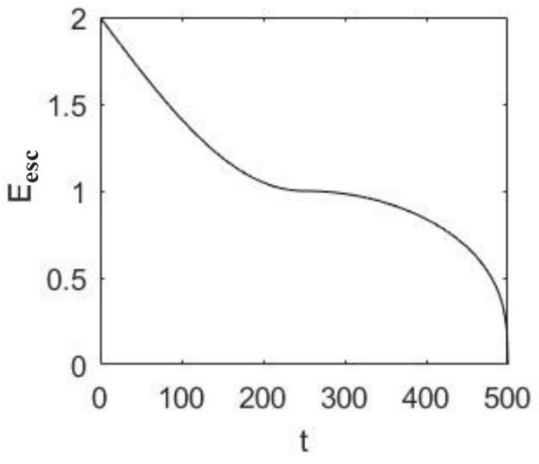

Fig 3. Recurrent alteration graph of Eesc.

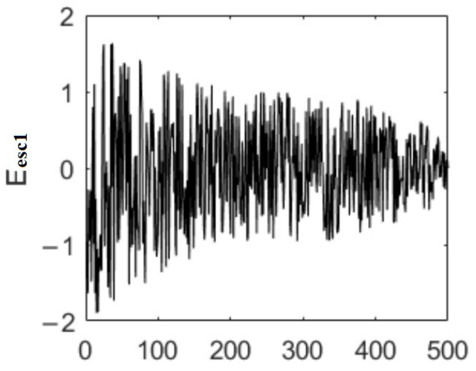

Fig 4. Recurrent alteration graph of Eesc1.

3.4.4 Gaussian Walk Learning

A traditional stochastic walk technique with good exploitation potential is Gaussian walk learning (GWL). Therefore, this research employs this method to modify population members to improve the population's diversity and assist it in escaping the local optimum trap. Equation (30) illustrates the Gaussian walk learning model:

$$Z(t+1) = Gauss(Z(t), \tau) \tag{30}$$

$$\tau = \cos\left(\pi/2 \times (t/T)^2\right) \times \left(Z(t) - Z_r\,(t)\right) \tag{31}$$

where the unidentified individual's position among the generation population t is shown by $Z(t)$, the individual in the generation population t is showed by $Z(t)$, and the Gaussian distribution with $Z(t)$ as the expectation and τ as the standard deviation is represented by $Gauss(Z(t), \tau)$. The function $\cos\,(\pi/2 \times (t/T)^2)$ modifies the stage dimension of Gaussian walk learning. Fig 5 illustrates this in image form. To improve the creation of algorithms and balance searchability, the disruption used during the beginning stages is larger and rapidly decreases in the latter phases.

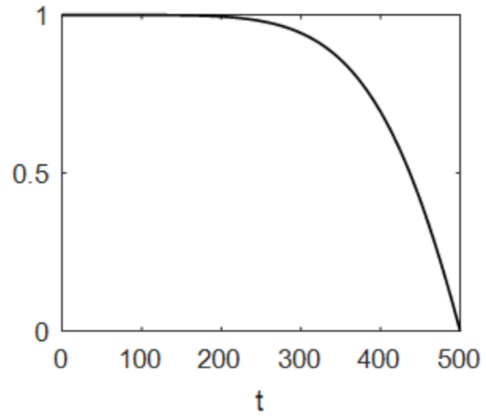

Fig 5. Wandering length of steps modification graph

3.4.5 MTHHO Algorithm

In overview, Algorithm 2 depicts the major stages of the enhanced MTHHO algorithm.

Algorithm 2 Principal phases of the MTHHO process

```
Input: Population size $N$ and the maximum number of iterations $T$
1: Initialise the population by Eq. (25)
2: while $t < T$ do
3: Determine the fitness of the original population and the people in its reverse population by generating the reverse population utilising the elite opposition-based learning process
4: If the process is not moving forward, then
5: Modify the location by Eq. (30).
6: else
7: for $i = 1{:}N$ do
8: Modify the escape energy $E_{esc}$ by Eq. (29)
9: if $|E_{esc}| \geq 1$ then
10: Modify the position by Eq. (12)
11: else if then $|E_{esc}| < 1$
12: if $|E_{esc}| \geq 0.5$ and $r \geq 0.5$ then
13: Modify the position by Eq. (15)
14: else if then $|E_{esc}| < 0.5$ and $r \geq 0.5$
15: Modify the position by Eq. (17)
16: else if then $|E_{esc}| \geq 0.5$ and $r < 0.5$
17: Modify the position by Eq. (21)
18: else if then $|E_{esc}| < 0.5$ and $r < 0.5$
19: Modify the position by Eq. (22)
20: end if
```

```
21: end if
22: end for
23: end if
24: $t = t + 1$
25: end while
26: return $Z_{prey}$
```

3.5 Proposed Scheduling Algorithm

The suggested approach aims to balance the impact on resources and minimise makespan and energy consumption by focusing on task dependencies and execution times. To maintain a balanced load on the resources, VMs are assigned tasks based on the requirements of the workflow tasks. Task assignment is a crucial aspect of a system, focusing on reducing bottlenecks and improving overall efficiency. It involves assigning tasks based on their dependencies and execution times, with higher-priority tasks given priority. Tasks are categorized based on thresholds for task levels and durations, ensuring timely handling of urgent tasks. The dynamic assignment process adjusts based on workload and resource availability, ensuring efficient use of virtual machines (VMs). The suggested method scans the workflow tasks and establishes thresholds for the tasks' depth and length, where depth and length are related to the tasks' levels and execution times, respectively. During task execution, the threshold standards are utilized to handle tasks based on various significances. Longer implementation times and bottlenecks in the system are caused by tasks with additional dependents. Likewise, to shorten the total execution time, tasks that take longer to complete must be prioritised. High-priority jobs are handled by allocating VMs with significant processing power to them. To prevent needless waiting for jobs at the same level, the suggested algorithm further looks for tasks with lengthy implementation periods and gives these tasks significance processing. Based on the input data, thresholds are established for the quantity of dependents as well as the duration. By using resources more effectively, these processes shorten the implementation period, which also results in lower energy consumption. Algorithm 3 illustrates the stages in the process.

Algorithm 3: Prevent bottleneck tasks

```
Input: workflow $w$
Output: Task queues according to length and depth
Allocate thresholds $dt$ for the depth of tasks and $lt$ for the length of tasks
for every task $t$ in the task set do
    depth = number of levels reliant on $t$
    length = execution time of $t$
     if $depth >= dt$ then
         transfer $t$ to the depth queue
     end
     if $length >= lt$ then
         transfer $t$ to length queue
     end
end
Return queues
```

Subsequently, the suggested algorithm employs queues based on task intensities; that is, distinct queues are kept for activities requiring a lot of CPU power and tasks with several dependencies. It takes less time to discover the right VMs during the VM allocation process when workloads with varying intensities are placed in distinct queues. The allocation process is streamlined by using queues to organise tasks, sorting and prioritising them before being assigned to VMs, reducing computational complexity and improving overall system efficiency. Tasks are stored in queues in which task queues are organised based on intensities, arrival times, and deadlines, with different queues maintained for tasks of varying intensities. Each task in the queue is associated with additional metadata, such as its intensity, arrival time, and deadline, which optimises the scheduling process. The last stage is to build appropriate VMs for the jobs after grouping them into distinct queues. Historical Scheduling Logs (HSLs) were used to inform the queue management process, providing insights into past task execution patterns. If no matching records exist, new VMs are created to handle the tasks, allowing the system to adapt to new or unexpected workloads. A novel VM is built using the resources essential to finish the task, and the HSLs are modernised if there isn't a matching record in the HSLs. Algorithms 4-6 illustrate the steps in the suggested algorithm.

Algorithm 4: Construct the types of VMs

```
Input: HSLs, task sets (from certain queues),
Output: VMs (Types)
$N$ = number of tasks in a queue
for (every task $t$ in $N$) do
    calculate $P(T_t)$ (Eq. 32)
    $H = best\ n\ P(T_t)$
end
for (every $l$ in $H$) do
    if ($t$ identified in HSLs) then
        assign $VM_j$ based on the type of $t$
        else
            Construct $VM()$
        end
        Allocate $VM_j$ to task $t$
    end
end
```

Algorithm 5: Construct Virtual Machines

```
Input: Set of tasks $L_t$ from queues from algorithm 3, set of hosts $L_h$
Output: Set of VMs
for (every host $h$ in $L_h$) do
    $u$ = resource consumption of $h$
    if (host resources are accessible) then
        Construct VM
        Modernise HSLs
    end
    else
        Turn on the new host
        Construct VM
        Modernise HSL
    end
end
```

Algorithm 6: Task Scheduling

```
Input: Set of tasks $L_t$ from queues from algorithm 3, set of VMs $V_l$
Output: Schedule of tasks for VMs
for (every task $t$ in $L_t$) do
    Sort the tasks into categories such as memory-intensive, CPU-intensive, or both.
    Set every category in a different queue.
    $V_l = VMtypes()$
    for (each VM $v$ in $V_l$) do
        Compute the same degree of $v$ and $t$
        if (the required condition is met) then
            schedule $t$ to $v$
            else
                $V_n = CreateVM()$
                schedule $t$ to $V_n$
                modernise HSLs
            end
        end
    end
end
```

The tasks are noted in the first phase, which means that suitable VMs for the tasks are located. The VM types are decided upon and the task types are categorised appropriately. Let T_l be a type l task for a set of tasks T that were taken from the historical data. Equation (32) can be used to calculate the ratio P.

$$P = \left|\frac{T_l}{T}\right| \tag{32}$$

Let T_i^r be a candidate task, and let V_j^r be a VM with $r = \{1,2,3,4\}$ denoting the CPU, memory, bandwidth, and storage capacity of the VM, respectively. And let $r = \{1,2,3,4\}$ be the task's requirements. Equation (33), when applied to task T_i^r and VM V_j^r, yields the matching degree.

$$P\left(T_i^r \middle| V_j^r\right) = \begin{cases} \left(V_j^r / T_i^r\right)^2, & if\ T_i^r > V_j^r \\ V_{max}^r - V_j^r + T_i^r / V_{max,}^r & otherwise \end{cases}, \tag{33}$$

where k denotes the type of VM and $V_{max}^r = k\epsilon U^{max} V_k^r$. Equation (34) can be used to determine the likelihood that a job T_j is of type Y_j.

$$P\left(Y_j \middle| T_i\right) = \pi_{r=1}^4 P\left(T_i^r \middle| V_j^r\right) \tag{34}$$

The suggested algorithm optimises the scheduling problem by utilising MTHHO. It is a difficult and complex task to map resources to tasks when there are several objectives. In this case, the dimension of the search field is calculated by the total amount of tasks in the process. The search space's size is adjusted to match the quantity of tasks. The dimensions' values range from 1 to the amount of VMs, depending on that number. The plotting of tasks to VMs in this research is represented by the symbols from earlier research i.e., $x_t^i = (x_t^{i\,1}, x_t^{i\,2}, \ldots, x_t^{i\,j})$, where $x_t^{i\,j}$ represents that, at time t, the j^{th} place of a particle is allocated to VM_i. The size of the search field is represented by the total amount of tasks within the process. The velocity is signified by $v_t^i = (v_t^{i1}, v_t^{i2}, \ldots, v_t^{ij})$, where v_t^{ij} $r$epresents the velocity, which represents that, at time t, VM_i transfers to the j^{th} place of a particle with velocity v. The method finds non-dominated solutions in subsequent iterations. The archive contains these solutions. We refer to these alternatives as workable solutions. Given that the processes identify non-dominated results, solutions are kept initially, and the record is empty. Only when the novel results outweigh the existing ones are they uploaded to the archive. Using the fitness function, the solutions' dominance is computed. Lastly, the archive only includes what are known as non-dominated, or viable, solutions.

4. Result And Discussion

This sector included the outcomes, performance measures, and comparative analysis of the suggested technique. The py-sim tool was used to implement the suggested heuristic-based enhanced Multi-Tactic Harris Hawks Optimization (MTHHO) algorithm on a 64-

bit Windows 10 Pro computer equipped with an Intel CoreTM i7-55000U CPU running at 2.40 GHz and 8 GB of RAM. The proposed method assumes that there will be 16 virtual machines (VMs) and 150 tasks in the cloud. The following factors led to the selection of the task and VM.

- Efficiency of Resources: The optimised resource allocation made possible by the assumption of 150 jobs and 16 resources (VMs) ensures that every task has enough processing power without wasting any of it.
- Predictable Performance: A steady task and resource count makes it easier to forecast and maintain the system's performance, which leads to reliable and strong cloud computing services.
- Simplified Management: Using a preset set of tasks and resources simplifies system management and makes it easier to scale resources dynamically in response to shifting workloads.

To determine which model receives a higher assessment score, tests will be conducted on both the proposed model and each experimented model. Our suggested method has been compared to several optimisation and hybrid methods to evaluate its possibility. Several configurations of the models have been tried to achieve the utmost basic TWT, TFT, cost, energy efficiency, and resource utilisation. Numerous examinations have been completed with the most comforting scheduling computations by this effort. To compare our model with other hybrid algorithms and outperform the communication scheduling issue in the cloud, this research has used enhanced optimisation. As a result, the suggested model has been improved. In this examination, a diversity of tasks and virtual machines were employed. When scheduling, every model demonstrates its capabilities.

4.1 Metrics and Parameters

The computational metrics listed beneath are utilised in this study to validate the outcomes of the suggested methodologies with other models.

Total Waiting Time, or TWT: This is a criterion that the user wants. When multiple resources vie for a single resource, it is the wait time for job execution. This is the amount of time that is spent waiting for an errand or cycle to finish its queue.

Total Finish Time, or TFT: This is a criterion that the user wants. It is the amount of time that passes according to plan from the start of an assignment until its completion. This is the point at which a task reaches its completion of execution.

Resource utilisation is a desired criterion as stated by the service provider. A further metric that shows the amplification of assets employed is resource utilisation. Although providers must use a certain number of resources to achieve maximum profit, resource utilisation should be high in the scheduling framework. One of the key implications in task scheduling is this parameter. There will be constant use of the resource. Energy efficiency and throughput are also very important; nevertheless, resource utilisation is another important barrier to task execution.

The amount of work that a process completes in a given amount of time is called throughput. In other words, throughput is the number of cycles over jobs finished in a given amount of period. The schedule ought to aim to increase the number of tasks completed in each time interval.

Energy efficiency: The amount of power used to process each client's request is known as energy consumption. A significant reduction in power consumption is required to achieve energy efficiency. This is one of the most important things to think about while trying to create an improved environment.

The suggested heuristic-based enhanced MTHHO method includes an initial evaluation of fitness values within the first population as shown in Fig 6. The best fitness values across 100 iterations are determined by the method using a round-robin technique based on computing time. The suggested method arrived at the lower fitness values of 33.338s, which decreased the makespan, by comparing the best fitness value with the previous fitness value in each iteration.

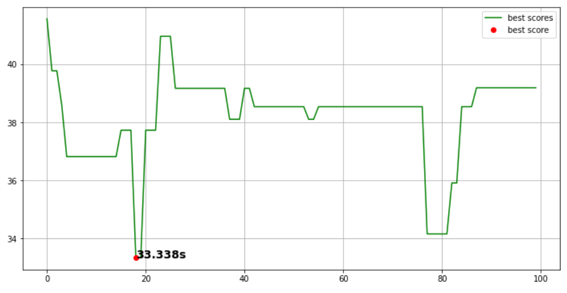

Fig 6. The proposed method Output for Best Fit algorithm

In this situation, the number of tasks (Nt) is kept constant at 500. Nonetheless, there is a 40-step variation in the number of virtual machines (Nv) between

40 and 200 VMs. The algorithms' relative performances are compared in terms of Makespan in Fig 7. As the number of machines increases in Fig 7, it is projected that the Makespan will decrease. For all scenarios from Nt= 40 to 200, the suggested approach outperforms all other algorithms. Additionally, it is evident that SJFP consistently has the highest MS values. The numerical results for Makespan are listed in Table 8 correspondingly.

Table 1. Makespan (MS) comparison of the proposed method ith the prior method

Nt	Algorithm									
	MCT	SJFP	LJFP	MinMin	MaxMin	PSO	SJFP-PSO	LJFP-PSO	MCT-PSO	Proposed
40	25.9	29.0	27.6	27.5	25.1	20.4	20.6	22.0	25.9	19.8
80	10.6	16.4	14.3	14.8	11.8	14.9	16.4	13.9	9.7	9.6
120	5.5	10.8	8.5	7.8	6.1	9.3	10.5	8.5	4.9	4.5
160	3.7	8.5	6.2	4.9	4.1	7.1	7.8	6.2	3.7	2.8
200	2.7	5.6	4.5	3.7	3.2	7.2	5.6	4.5	2.7	2.4

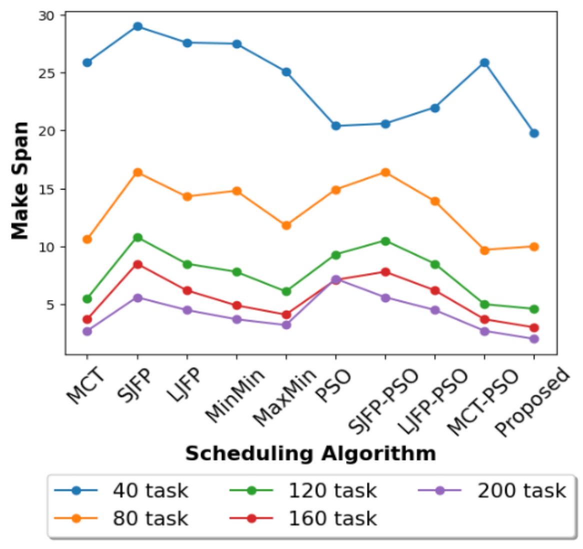

Fig 7. Performance evaluation among processes using makespan (MS).

To contrast the proposed heuristic-based enhanced MTHHO method's QoS performance metrics with those of earlier techniques such as round robin (RR), PSO, first come first serve (FCFS), genetic simulated annealing (GASA), and shortest job first (SJF), HGA (Hybrid Genetic Algorithm).

Table 2. Total outcomes across various approaches.

Parameters	SJF	FCFS	RR	PSO	GASA	HGA	Proposed
Total execution time	55.36	54.68	54.31	40.21	36.22	32.57	16.75
Total finish time	101.67	100.18	99.31	80.10	79.4	76.6	03.15
Throughput	0.72	0 .73	0.74	1.01	0.99	1.21	2.4
Resource utilization	0.42	0.42	0.4	0.61	0.63	0.69	0.95
Energy efficiency	0.60	0.62	0.55	0.35	—	0.30	0.20

Fig 8. shows the relationship between all employed strategies and the TFT and TET, which are some of the validation criteria used to verify the efficacy of the suggested technique. These are carefully taken into account when making plans to increase QoS. The results show that we may attain the highest level of resource utilisation. Every activity in RR receives the same amount of time, but as the findings show, there are several situations in which an average waiting time could be problematic. Similar data was used to assess the result and examine how the calculation was presented. Despite its speed advantage, the conventional method has the longest waiting period next streamlining; the other AI models that were tested

were likewise effective, but not as effective as the suggested model. Consequently, our suggested model outperforms all other methods, which is our benchmark. Furthermore, in comparison to other strategies, the suggested method offers the shortest execution time, resulting in a faster task performance. So that users can avoid task terminations, the waiting time should be as short as possible. This can determine whether a task is reasonable and the best method to employ when arranging a cloud-based scheduling procedure. On the other hand, while some models have a longer completion time, our model outperforms others in terms of finishing period.

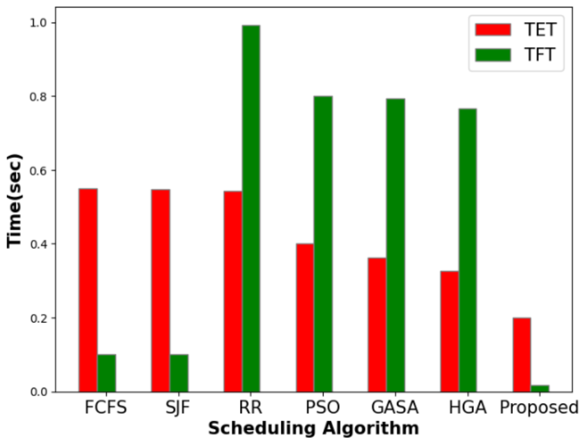

Fig 8. TET, and TFT of the scheduling model.

The suggested model with the highest throughput is the best technique, as demonstrated by the throughput relationship between each approach and Fig 9. Following a sequence of numerous tasks, efforts were prepared to increase the throughput. One of the most important factors in demonstrating the existence of a cycle for every time unit will undoubtedly be throughput. The throughput result demonstrates the effectiveness of the suggested model. To represent the exhibition, each assignment was broken into ten parts. The suggested method executes superior to further methods in this scenario during the split. Despite being linearly separable, the optimisation strategies demonstrated their effectiveness. In any case, the suggested model was the superior one. It may result that the suggested method outperforms previous methods and meets this depiction well because the duration is the largest number of tasks that can be finished per time unit as shown in Fig 10.

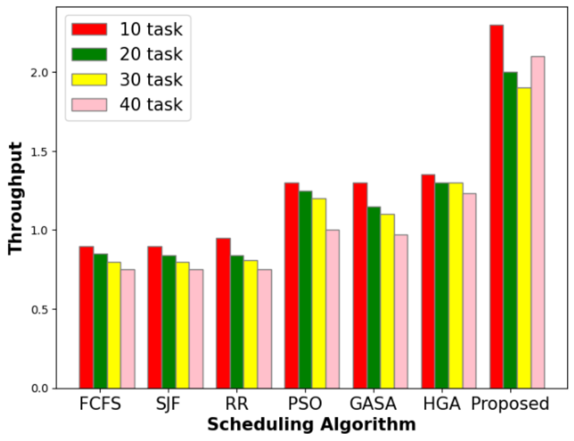

Fig 9. Result of the throughput for different tasks.

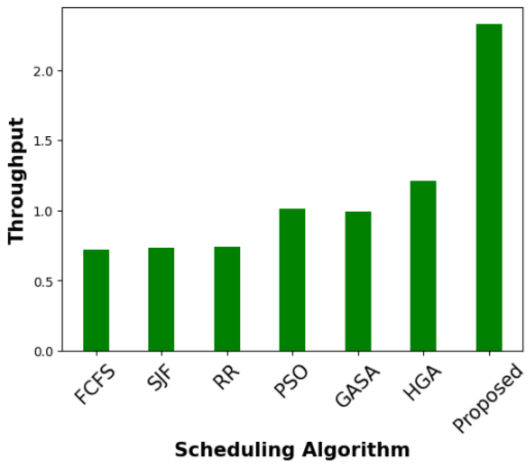

Fig 10. Comparison of the throughput

Fig 11. displays how the scheduling strategies' resource utilisation relates to each other. Additionally, the suggested model selects a different request and makes use of the resources that are available during runtime. In comparison to alternative methods, the suggested computation reduces the inactive waiting time in this way. Similarly, asset utilisation is enhanced independently. However, some assets can become excellent when they can be used in combination with others. The resource is examined under several different makespan totals. By increasing the amount of resources used, the approaches maintain their regular state. The typical waiting time often increases with resource size or the number of task increments. It follows that, in comparison to the other contrasting strategies, the suggested model is the most effective. The effectiveness of different techniques in comparison to the suggested technique may be inferred from the Fig The normal asset utilised by diverse procedures is practically equivalent, indicating that the number of available resources has an impact on it.

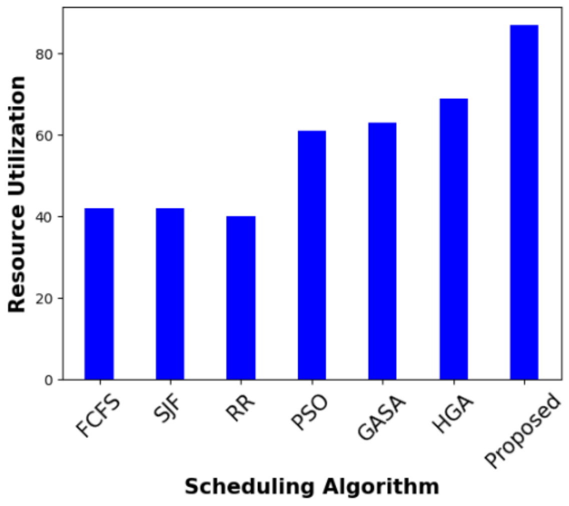

Fig 11. Scheduling models vs resource utilization.

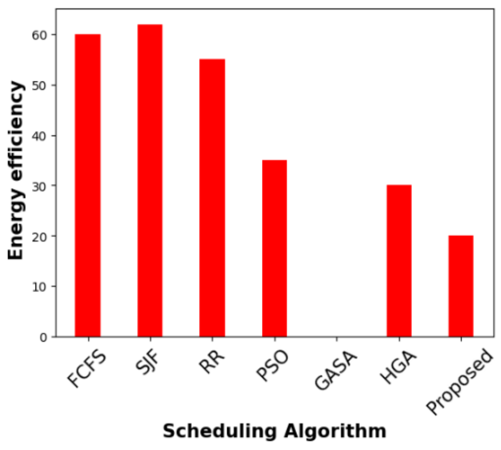

Fig 12. Efficiency of energy consumption comparison with the proposed model.

The efficiency of energy usage is shown in Fig 12. To fully examine its efficiency, the energy was first computed in KWh and then converted to a percentage. The parameter's goal is to lower energy consumption. The suggested approach beats further methods using a 20% lower amount of energy consumption, as the Fig illustrates. The suggested model's efficiency was demonstrated by a comparison with alternative metaheuristics. In comparison, the PSO and GA appeared quite equitable; nonetheless, the model's efficiency in energy usage is still lacking. Being one of the important criteria, this one will improve machine performance while also promoting environmental sustainability. The standard conventional procedure that has been tested is renowned for its sufficiency, however, it was unable to outperform the presented model.

5. Conclusions

In conclusion, the paper presents a comprehensive approach aimed at optimizing VM allocation in cloud data centres, addressing challenges related to energy efficiency and task dependencies. By employing an enhanced MTHHO algorithm, this approach effectively assigns energy-efficient VMs considering job dependencies and task execution times. The algorithm incorporates various improvements, including enhanced energy updating methods, elite opposition-based learning for flexibility, Sobol sequences for population initialization, and the introduction of Gaussian walk learning to prevent the process from converging to local optima. The results demonstrate significant enhancements in Quality of Service (QoS) performances, showcasing reduced makespan, energy consumption of 0.20, throughput of 2.4, and execution time of 16.75, with resource allocation improvements ranging from 1% to 98% compared to prior methods of cloud computing. Overall, the proposed heuristic-based MTHHO method efficiently balances loads and allocates resources, highlighting its potential for enhancing cloud computing efficiency and performance.

Reference

Abd Elaziz, M., Xiong, S., Jayasena, K. P. N., & Li, L. (2019). Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. Knowledge-Based Systems, 169, 39-52.

Abdullahi, M., & Ngadi, M. A. (2016). Symbiotic organism search optimization-based task scheduling in a cloud computing environment. Future Generation Computer Systems, 56, 640-650.

Abdullahi, M., Ngadi, M. A., Dishing, S. I., & Abdulhamid, S. I. M. (2023). Adaptive symbiotic organisms search for constrained task scheduling in cloud computing. Journal of ambient intelligence and humanized computing, 14(7), 8839-8850.

Adhikari, M., Nandy, S., & Amgoth, T. (2019). Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud. Journal of Network and Computer Applications, 128, 64-77.

Agarwal, M., & Srivastava, G. M. S. (2021). Opposition-based learning inspired particle swarm optimization (OPSO) scheme for task scheduling problems in cloud computing. Journal of Ambient Intelligence and Humanized Computing, 12(10), 9855-9875.

Alam, T. (2021). Cloud-based IoT applications and their roles in smart cities. Smart Cities, 4(3), 1196-1219.

Alsaidy, S. A., Abbood, A. D., & Sahib, M. A. (2022). Heuristic initialization of PSO task scheduling algorithm in cloud computing. Journal of King Saud University-Computer and Information Sciences, 34(6), 2370-2382.

Chaudhary, D., & Kumar, B. (2018). Cloudy GSA for load scheduling in cloud computing. Applied Soft Computing, 71, 861-871.

Cui, D., Peng, Z., Li, Q., He, J., Zheng, L., & Yuan, Y. (2021). A survey on cloud workflow collaborative adaptive scheduling. In Advances in Computer, Communication and Computational Sciences: Proceedings of IC4S 2019 (pp. 121-129). Springer Singapore.

Devaraj, A. F. S., Elhoseny, M., Dhanasekaran, S., Lydia, E. L., & Shankar, K. (2020). Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments. Journal of Parallel and Distributed Computing, 142, 36-45.

Fanian, F., Bardsiri, V. K., & Shokouhifar, M. (2018). A new task scheduling algorithm using Firefly and simulated annealing algorithms in cloud computing. International Journal of Advanced Computer Science and Applications, 9(2).

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.

Golchi, M. M., Saraeian, S., & Heydari, M. (2019). A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: Performance evaluation. Computer Networks, 162, 106860.

Jena, U. K., Das, P. K., & Kabat, M. R. (2022). Hybridization of a meta-heuristic algorithm for load balancing in a cloud computing environment. Journal of King Saud University-Computer and Information Sciences, 34(6), 2332-2342.

Katal, A., Dahiya, S., & Choudhury, T. (2023). Energy efficiency in cloud computing data centres: a survey on software technologies. Cluster Computing, 26(3), 1845-1875.

Keshanchi, B., Souri, A., & Navimipour, N. J. (2017). An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing. Journal of Systems and Software, 124, 1-21.

Konjaang, J. K., & Xu, L. (2021). Meta-heuristic approaches for effective scheduling in infrastructure as a service cloud: A systematic review. Journal of Network and Systems Management, 29, 1-57.

Kumar, M., & Sharma, S. C. (2018). PSO-COGENT: Cost and energy-efficient scheduling in a cloud environment with deadline constraints. Sustainable Computing: Informatics and Systems, 19, 147-164.

Mahato, D. P., Singh, R. S., Tripathi, A. K., & Maurya, A. K. (2017). On scheduling transactions in a grid processing system considering load through ant colony optimization. Applied Soft Computing, 61, 875-891.

Mansouri, N., Zade, B. M. H., & Javidi, M. M. (2019). Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory. Computers & Industrial Engineering, 130, 597-633.

Mishra, S. K., Sahoo, B., & Parida, P. P. (2020). Load balancing in cloud computing: a big picture. Journal of King Saud University-Computer and Information Sciences, 32(2), 149-158.

Ramamoorthy, S., Ravikumar, G., Saravana Balaji, B., Balakrishnan, S., & Venkatachalam, K. (2021). MCAMO: multi-constraint aware multi-objective resource scheduling optimization technique for cloud infrastructure services. Journal of Ambient Intelligence and Humanized Computing, 12, 5909-5916.

Strumberger, I., Bacanin, N., Tuba, M., & Tuba, E. (2019). Resource scheduling in cloud computing based on a hybridized whale optimization algorithm. Applied Sciences, 9(22), 4893.

Wei, X. (2020). Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing. Journal of Ambient Intelligence and Humanized Computing, 1-12.

Yadav, M., & Mishra, A. (2023). An enhanced ordinal optimization with lower scheduling overhead based novel approach for task scheduling in a cloud computing environment. Journal of Cloud Computing, 12(1), 8.