Journal of Algorithms and Computation

journal homepage: http://jac.ut.ac.ir

Priority-Oriented Task Scheduling based on Harris Hawks Optimizer for Cloud Computing

Reyhane Ghafari*[1] and Najme Mansouri†[1]

[1]Department of Computer Science, Shahid Bahonar University of kerman, Kerman, Iran.

ABSTRACT

Cloud computing is a high-performance computing environment that can remotely provide services to customers using a pay-per-use model. The principal challenge in cloud computing is task scheduling, in which tasks must be effectively allocated to resources. The mapping of cloud resources to customer requests (tasks) is a popular Nondeterministic Polynomial-time (NP)-Complete problem. Although the task scheduling problem is a multi-objective optimization problem, most task scheduling algorithms cannot provide an effective trade-off between makespan, resource utilization, and energy consumption. Therefore, this study introduces a Priority-based task scheduling algorithm using Harris Hawks Optimizer (HHO) which is entitled as PHHO. The proposed algorithm first prioritizes tasks using a hierarchical process based on length and memory. Then,

Keyword: Cloud, Task scheduling, Meta-heuristic, Task priority.

AMS subject Classification: 05C78.

ARTICLE INFO

Article history:
Research paper
Received 12, October 2021
Received in revised form 14, November 2021
Accepted 29 November 2021
Available online 30, December 2021

*reihaneh.ghafary@gmail.com
†Corresponding author: N. Mansouri. Email: najme.mansouri@gmail.com

1 Abstract continued

the HHO algorithm is used for optimally assigning tasks to resources. The PHHO algorithm aims to decrease makespan and energy consumption while increasing resource utilization and throughput. To evaluate the effectiveness of the PHHO algorithm, it is compared with other well-known meta-heuristic algorithms such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Whale Optimization Algorithm (WOA), Salp Swarm Algorithm (SSA), and Moth-Flame Optimization (MFO). The experimental results show the effectiveness of the PHHO algorithm compared to other algorithms in terms of makespan, resource utilization, throughput, and energy consumption.

2 Introduction

Cloud computing is a distributed computing paradigm that provides services for users around the world. Users can access the cloud computing environment through the cloud user ID remotely or locally [44]. Task scheduling is an NP-Complete problem in the cloud environment. To solve this problem, meta-heuristic algorithms can be used to decrease the polynomial complexity [24]. Task scheduling with priority attention is one of the main challenging issues in the cloud environment that can increase user and service provider satisfaction. It can also achieve efficient utilization of resources with maximum profit [3]. To address this challenge, this paper presents a novel task scheduling algorithm that first uses the Analytic Hierarchy Process (AHP) to prioritize tasks before sending them to the scheduler. Then, the HHO algorithm is applied to improve task scheduling behavior by taking into account parameters such as makespan, throughput, resource utilization, and energy consumption.

2.1 Cloud Computing

Cloud computing has become a major part of modern Internet technologies due to its unique feature [26],[28]. It consists of large and power-consuming data centers to provide reconfigurable computing resources. Figure 1 shows various definitions of cloud computing in various studies.

It can be seen that cloud computing has various definitions. However, one of the most famous definitions of cloud computing is the National Institute of Standards and Technology (NIST) definition [30].The NIST's definition is summarized in Fig. 2. The cloud computing system provides on-demand computing and storage services with excellent properties like reliability, scalability, pay-as-you-go pricing model, and reliability. As the number of users and demands increased, various companies (e.g., Google Compute Engine, Amazon EC2) began to offer software and hardware to users. Cloud Service Providers (CSPs) provide various kinds of services however, services are generally categorized into three main categories (i.e., Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS),

computing.png

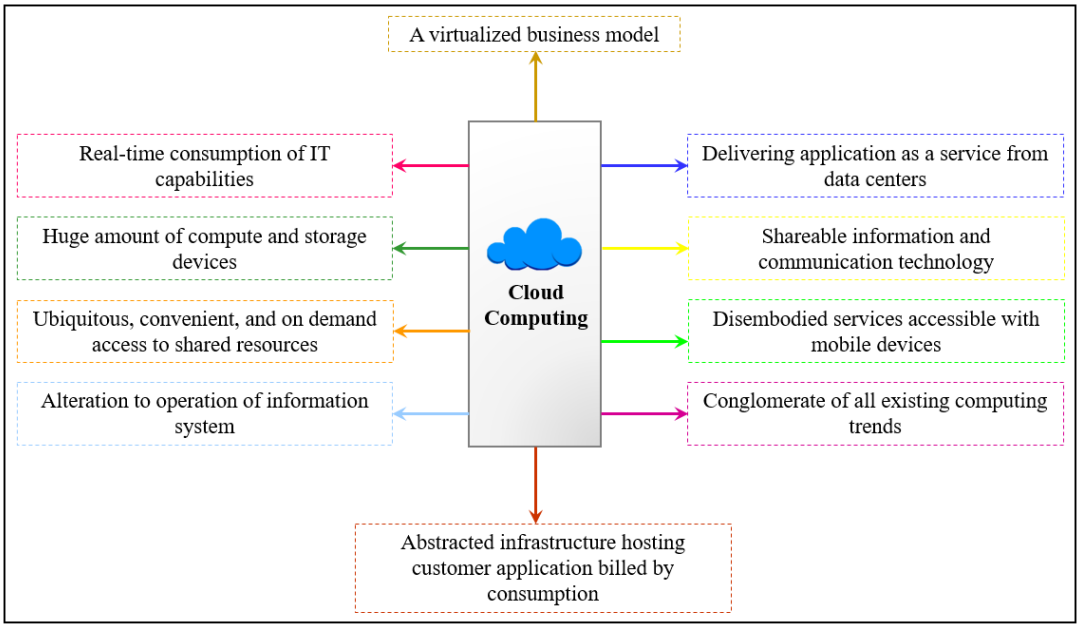

Figure 1: Definitions of cloud computing [5].

and Infrastructure-as-a-Service (IaaS) that are provided based on the needs and demands of users [38].
Cloud computing manages different types of virtual resources that make scheduling an important component. In the cloud, a user may use several thousand virtual assets for each task. Optimal allocation of tasks to resources can improve system performance and efficiency and so play a key role in the cloud environment [61].

2.2 Task Scheduling

Cloud computing not only helps different types of applications but also creates virtual conditions for applications to run efficiently. An appropriate task scheduler is required to arrange the execution of tasks in a cloud system [19]. The task scheduler must utilize the cloud resources to execute the tasks. In recent years, researchers have paid close attention to the problem of task scheduling in the cloud system [56]. In a cloud environment, users send their requests to the task manager component. The scheduler assigns the requests (which are received from the task manager) to the appropriate Virtual Machines (VMs) by utilization of suitable task scheduling algorithms to increase the efficiency of the cloud system (available resources are analyzed using a resource information server). If the task scheduling algorithm is efficient, it improves resource utilization and increases user and CSP satisfaction. Figure 3 shows the problem of task scheduling in the cloud.
Prioritizing tasks is a significant issue in task scheduling because some tasks that cannot stay in the system for long periods need to be serviced earlier than others. In other

de.png

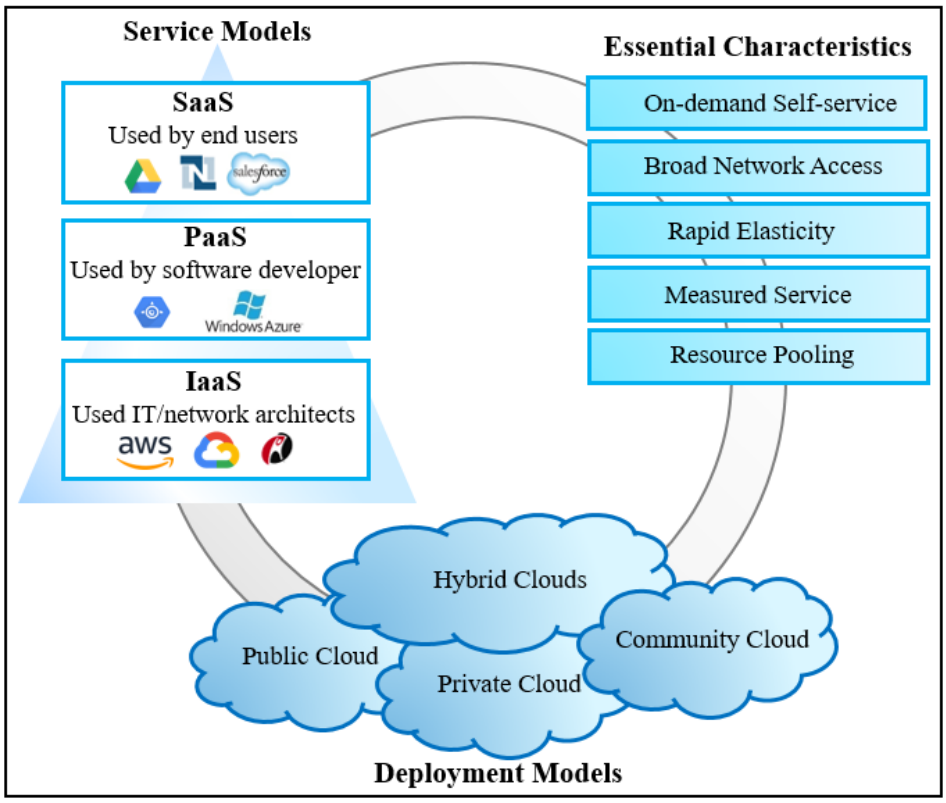

Figure 2: NISTs definitions of cloud computing.

Sch.png

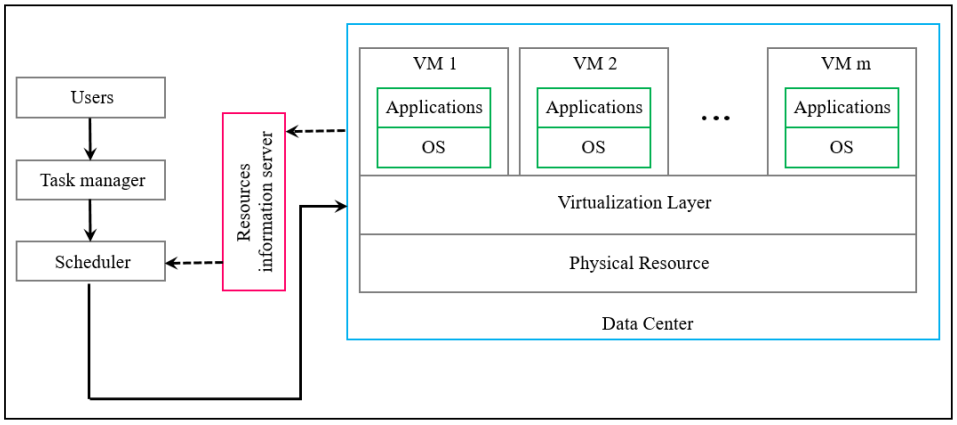

Figure 3: Task scheduling framework in the cloud [11].

words, priority indicates the urgency of a task to be completed as soon as possible. An appropriate task scheduling algorithm should pay attention to the priority of tasks [51]. Priority can be set based on various criteria (e.g., deadline of a task, length of a task, arrival time of a task, or memory size of a task). Since in cloud, there is a wide range of features that must be considered, so a proper task scheduling algorithm in the cloud

should pay attention to multi-attribute and multi-criteria features of tasks [37]. There are various Multi-Criteria Decision-Making (MCDM) models that are based on mathematical modeling. Based on a pairwise comparison according to the MCDM model, a model called AHP was proposed in 1980 by Thomas Saaty [42]. AHP has been used in various fields over the past few decades. AHP is an appropriate model for priority-based task scheduling problems. In addition, task scheduling is a significant problem in cloud computing by taking into account various factors such as makespan, energy consumption, completion time, cost, resource utilization, and throughput. A good scheduler should execute tasks with fewer resources and in a shorter time [43]. Using fewer resources means consuming less energy. One of the most important matters in cloud environments is to minimize makespan and energy consumption. The process of detecting the appropriate solution according to conflicting criteria such as energy consumption and makespan by a parallel application with priority limitation is a multi-objective problem. The solution to this type of problem is a set of Pareto points (those that improvement in one goal can only happen with the worsening of at least one other goal) [52]. Meta-heuristic algorithms have been suggested as non-deterministic methods to solve scheduling problems in a polynomial time [46].

2.3 Meta-heuristic Algorithm

One of the major challenges in the cloud system is optimal task scheduling. Various optimization techniques like heuristic techniques are used to solve the scheduling problem [27]. As the complexity of the problem increases, heuristic techniques have achieved very restricted success between different applications. This restriction is due to the delay in reaching the optimal solution. Therefore, heuristic techniques do not have the necessary efficiency. On the other hand, there are meta-heuristic algorithms that are expected to overcome these restrictions and provide the best solution in a shorter time [20]. Meta-heuristic scheduling algorithms provide better scheduling outcomes than traditional and heuristic algorithms. Figure 4 shows a schematic diagram for executing task scheduling algorithms based on meta-heuristic algorithms.
Meta-heuristic algorithms have become very popular in recent years because of their efficiency in solving large and complex problems. Meta-heuristic strategies have many beneficial properties, such as the following [8]:

- Meta-heuristic algorithms are not problem-dependent.
- These algorithms effectively explore the search space so that they can find a near-optimal solution to solve NP-complete problems.
- Meta-heuristic algorithms are usually approximate and non-deterministic.

Meta-heuristic algorithms are executable to solve problems in various fields with very acceptable performance because they are independent of the problem to be solved. Generally, meta-heuristic algorithms have two types: i) single solution-based (processing only

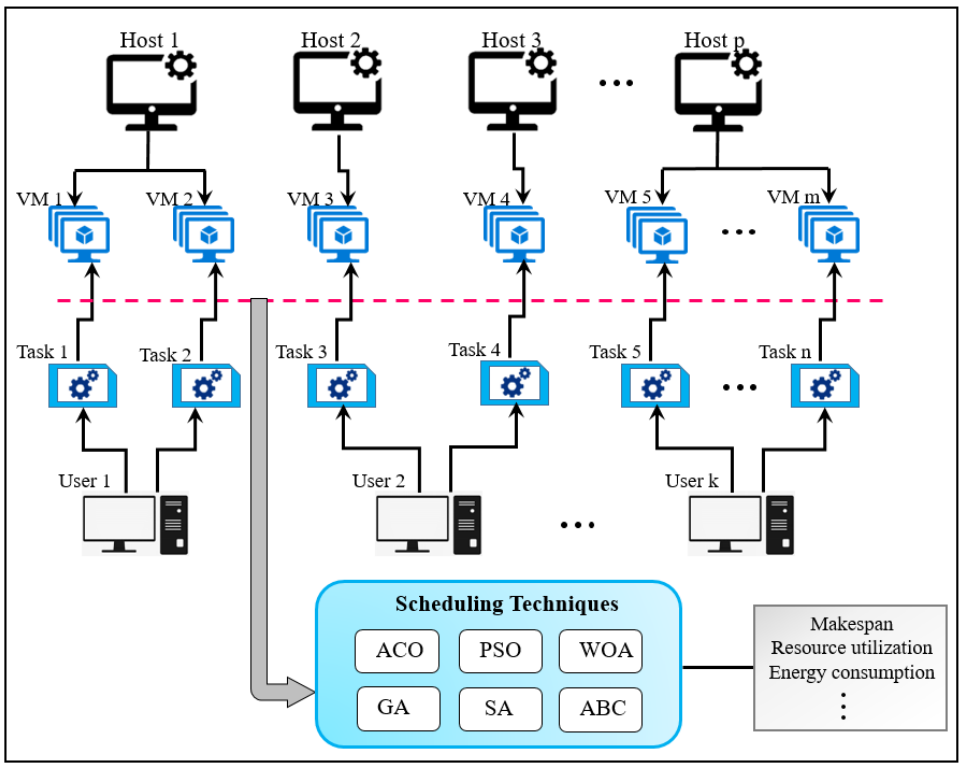

Figure 4: The schematic diagram for executing the scheduling algorithm [50].

one solution in the optimization phase) and ii) population-based (processing a set of solutions in each iteration) [54]. The population-based strategies usually discover an optimal or near-optimal. In the population-based strategies, the optimization process begins with the creation of the initial population (each individual indicates a candidate solution), then the population is updated by using some random operators and the optimization process continues until the stop condition is reached [16]. As shown in Fig. 5, the population-based algorithms can be categorized into four principal categories: evolution-based methods that are inspired by the laws of natural evolution, physics-based methods that are inspired by the physical laws in the world, human-based methods that are inspired by advancement in the level of searching strategy, and swarm-based methods that are expanded based on mathematical models inspired by the activities and cooperative behavior of various species like birds, ants, and bacteria, which live in groups and cooperate for searching and collecting food. Although there are differences between various meta-heuristic algorithms, they divide the search process into two stages of exploration and exploitation. In the exploration stage, the algorithm must use random operators to search the promising regions of the search space randomly and globally. In the exploitation stage, the algorithm must perform a local search in the promising areas achieved in the exploration stage. One of the main challenges of meta-heuristic algorithms is finding the right balance between exploration and exploitation [49].

One of the recent population-based optimization algorithms is the Harris Hawks Opti-

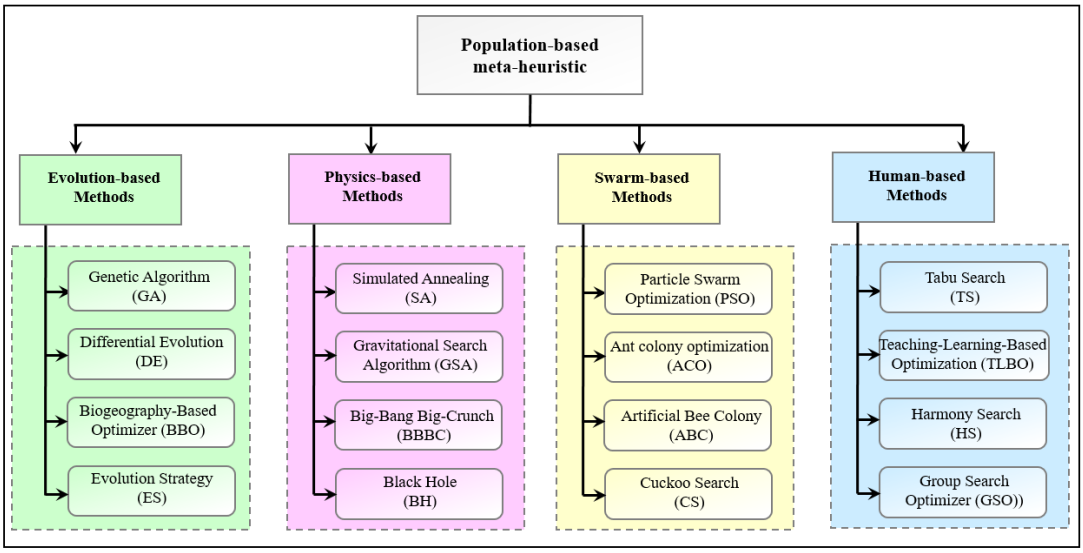

Figure 5: Categorization of the meta-heuristic algorithms [39].

mizer (HHO) [17]. HHO is inspired by the cooperative behavior and chasing style of Harris's hawks in nature, which is named surprise pounce. In this clever technique, several hawks work together to pounce a prey from various directions to surprise it. Harris hawks can show various types of chasing patterns according to the different scenarios and escaping approaches of the prey. Based on the results obtained in [17], the HHO algorithm have better performance compared to the PSO [23], GA [18], Biogeography-Based Optimizer (BBO) [48], Gray Wolf Optimization (GWO) [34], Differential Evolution (DE) [53], Cuckoo Search (CS) [59], Teaching Learning Based Optimization(TLBO) [40], BAT Algorithm (BA) [57], Flower Pollination Algorithm (FPA) [60], Moth-Flame Optimization (MFO) [31], and Firefly Algorithm (FA) [58]. The HHO algorithm can have a significant role in solving various real-world optimization problems (e.g., pattern recognition, geotechnical engineering, engineering design, optimization, feature selection, image segmentation, manufacturing optimization, power quality, and drug design). Moreover, the HHO can be used for problems with the unknown types of search space or problems containing discrete and continuous spaces, provide better quality solutions, provide high accuracy in extracting optimal parameters, and enhance the prediction performance. In addition, the results demonstrated that HHO is a powerful optimizer that aids to solve complex nonlinear problems and can find the optimal solution quicker [2]. The results showed that the main hawk starts the search operation with sudden movements and first increases the variety and explores the favorable regions of the solution space. The range of these changes covers more than 50% of the solution space. This indicates the exploration capability of HHO. Over time, the range of the fluctuations gradually reduces. This ensures the transfer of HHO from exploratory processes to exploitative stages. Finally, the first hawk movement pattern becomes very stable, indicating that HHO is exploiting promising areas in the final stages. Avoiding local optimization and a smooth transi-

tion from exploration to exploiting are the main benefits of the HHO algorithm. The results show that the HHO can reveal an accelerated convergence process. The main contributions of this paper are listed below:

- A priority-based task scheduler is proposed, which sorts user tasks by priority. The priority is computed using AHP based on tasks length and memory.
- A task scheduling problem is formulated and a mathematical model and objective functions are defined that optimally assign tasks to VMs.
- The multi-objective fitness function has been used concerning makespan, throughput, resource utilization as well as energy consumption.
- To improve the efficiency of tasks execution in a cloud environment, the HHO algorithm is used to solve the problem, which can find excellent solutions, easy to executed, and flexible.
- To confirm the effectiveness of the presented algorithm, the efficiency of the PHHO algorithm is compared with six popular meta-heuristic algorithms. Evaluation is performed using four objective criteria: makespan, average resource utilization, average throughput, and total energy consumption.

The rest of this paper is organized as follows: Section 3 explains some previous works related to the scheduling in the cloud, Section 5 presents a background of AHP and HHO, Section 6 describes the PHHO algorithm task scheduling algorithm, Section 7 represents the experimental result, and finally Section 8 discusses conclusion and future works.

3 Related Works

In recent years, many researchers have paid attention to many issues facing cloud computing. Task scheduling is one of the main problems in the cloud environment because it can have a huge impact on system performance. However, there is no exact optimal solution that optimizes all parameters in cloud scheduling. Some of the scheduling strategies are discussed below.

Shojafar et al. [45] proposed a hybrid algorithm named FUGE, which used GA [18] as the basis of their algorithm and modified it with the help of fuzzy theory for allocating jobs. The purpose of the presented algorithm is to decrease the makespan, cost, and degree imbalance in the cloud when scheduling tasks. In the FUGE algorithm, two kinds of chromosomes are created based on various Quality of Service (QoS) parameters. Then the fuzzy theory is used to calculate the fitness values of all chromosomes and for crossover operation. The presented algorithm allocates tasks to resources based on VM memory, job lengths, VM processing speed, and VM bandwidth. The experimental results showed that FUGE performed better than other algorithms in terms of execution time, execution cost, and average degree of imbalance. The experimental results also showed that FUGE

improved by about 45% in terms of execution cost and about 50% in terms of total execution time compared to GA. The main weakness of the FUGE algorithm is that it is not considered energy consumption.

Chen et al. [6] offered a WOA [33]-based multi-objective task scheduling algorithm that purposes to improve system performance based on given computational resources. The focus of the algorithm was to decrease the execution time, load, and price cost. To improve the efficiency of the WOA, the authors also suggested the improved WOA algorithm for assigning tasks to resources in the cloud environment and called it IWC. The experimental results showed that IWC has a better convergence speed and accuracy and also can perform better in system resource utilization for both small-scale and large-scale tasks compared to other meta-heuristic algorithms. However, the IWC algorithm is not considered constraints such as priority constraints.

Mansouri and Javidi [29] suggested a job scheduling algorithm based on the cost and named it CJS. The CJS simultaneously considered both types of jobs, data-intensive and computation-intensive. The presented algorithm uses data, processing power, and network features to assign tasks to resources. The scheduler chooses the best location according to the dynamic state of the network, the location of the data, the size of the data, and the pool of processing cycles. The simulation results showed that CJS performed better in terms of makespan and success rate than other algorithms. But, energy consumption is not considered.

Er-raji et al. [12] suggested a task scheduling algorithm to execute tasks efficiently. The authors improved tasks execution time using task length and VM processing speed, and the number of tasks per VM. The purpose of the proposed algorithm is to reduce the execution time of the tasks. The authors used CloudSim to evaluate the performance of the presented algorithm. The experimental results represented that the presented algorithm has better performance in terms of total execution time compared to other algorithms. The main weakness of the presented algorithm is that this algorithm focuses only on reducing execution time and is not considered other key QoS parameters such as energy consumption and resource utilization.

Jacob and Pradeep [21] offered a hybrid task scheduling algorithm called CPSO, which is a combination of CS [59] and PSO [23]. The goal of the presented algorithm is to decrease the makespan, cost, and deadline violation rate. The CPSO algorithm decreases all cost factors such as performance cost and user costs. The simulation results showed that the CPSO algorithm has better performance than other algorithms in terms of makespan, cost, deadline violation rate. In the proposed algorithm, the probability of resources overloading is high.

Kumar and Venkatesan [25] introduced an effective task scheduling algorithm called HGPSO. The proposed algorithm first stores the user's tasks in the queue manager, their priority is computed, and if it is a repeated task, it is assigned to the appropriate resource. Novel tasks are analyzed and stored in the on-demand queue. The HGPSO algorithm (a combination of the GA and PSO algorithms) gets the results of the on-demand queue. Then, the HGPSO assigns tasks of the on-demand queue to the appropriate resources. The experimental results showed that the HGPSO performs better in terms of execution

time, scalability, and availability compared to other algorithms. However, the energy efficiency of the HGPSO algorithm is very low.
Abd Elaziz et al. [1] suggested a new hybrid algorithm called MSDE, which is a combination of the Moth Search Algorithm (MSA) [55] and DE [53]. The purpose of the MSDE algorithm is to reduce the makespan. The MSA algorithm is inspired by the behavior of moths insects and their relationships. However, MSA's exploitation capability is not as good as its exploratory capability, so the authors used the DE algorithm as a local search strategy to enhance MSA exploitation capability. The experimental results showed that MSDE performs better in terms of makespan than other algorithms and can assign tasks to VMs effectively. The major weakness of the MSDE algorithm is that it focuses only on makespan and doesnt consider other QoS parameters like energy consumption and resource utilization.
Guo [15] introduced a multi-objective task scheduling algorithm based on the fuzzy self-defense algorithm. The author considered the shortest time that customers need to wait, the degree of resource load balance, and the cost as the objective function. Then, the optimal solution is determined based on fuzzy self-defense algorithm. The experimental results showed that the presented algorithm performed better in terms of maximum completion time, deadline violation rate, and resource utilization compared to other algorithms. But, the author didnt consider the energy consumption during the scheduling process.
Table 1 summarizes the discussed scheduling algorithms in terms of various parameters. An analysis of the related works in Table 1 represents that most traditional scheduling algorithms in the cloud environment focused on minimizing makespan regardless of energy consumption and priority constraint. Therefore, this paper introduces a novel algorithm to address these issues. The proposed algorithm first sorts the tasks using a hierarchical process and then optimally assigns the tasks to the resources using the HHO algorithm. The goal of the PHHO is to make a trade-off between four objectives (i.e., makespan, energy consumption, throughput, and resource utilization).

4 Background

This section describes the background of the AHP and the HHO algorithm.

4.1 Analytic Hierarchy Process (AHP)

AHP is a multi-criteria decision-making tool for dealing with complex decision problems [7], [36]. In the AHP method, it is not necessary to define a complex expert system and AHP decides based on the set of evaluation criteria and a set of alternative options. The hierarchical model includes three levels of goal, criteria, and alternatives [10]. Figure 8 shows the hierarchical structure of AHP.
The relative values of alternatives or criteria are determined by the Saaty Rating Scale [41]. Table 4 shows the AHP numerical scales, which vary from 1 to 9.

Table 1: Summary of discussed scheduling strategies.

Reference	Year	Makespan	Resource utilization	Security	Energy consumption	Priority constraint	Tool	Technique applied	Disadvantage
Shojafar et al. [45]	2015	+	-	-	-	-	CloudSim	Fuzzy theory and GA	- Does not consider VM migration and energy consumption, - The monitoring overhead is high.
Chen et al. [6]	2019	-	+	-	-	-	Matlab	Improved WOA	- Does not consider sufficient agents to search for the best resource, - As the workload increases, the scheduling overhead increases, - Does not perform well in convergence speed and accuracy.
Mansouri and Javidi [29]	2019	+	-	-	-	-	CloudSim	Data, processing power, and network characteristics in the job assignment process	- Does not pay attention to security aspect for business clouds, - Does not use optimization methods.
Er-raji et al. [12]	2018	+	-	-	-	+	CloudSim	Quicksort algorithm	- Does not take into account VM classification and task migration, - The algorithm focuses only on user satisfaction, - Does not consider the dependent tasks.
Jacob and Pradeep [21]	2019	+	-	-	-	-	CloudSim	PSO and CS	- Does not consider a priority, - Does not discuss the reliability or energy QoS parameters.
Kumar and Venkatesan [25]	2018	+	-	-	-	+	—	GA and PSO	- Energy consumption is high, -Does not take into account SLA violation and deadline.
Elaziz et al. [1]	2019	+	-	-	-	-	CloudSim	MSA and DE	- Time complexity of the algorithm is high, - The presented algorithm is single-objective and focuses only on makespan reduction and does not consider other QoS parameters such as resource usage, energy, reliability, etc.
Guo [15]	2021	-	+	-	-	-	—	Fuzzy self-defense	Energy consumption is high, - Does not consider priority constraint.

AHP provides a way to break down a problem into a hierarchy of sub-problems for easy evaluation. AHP steps are as follows:
Step 1) Divide the problem into hierarchies of goal, criteria, and alternatives.
Step 2) Collect data of the hierarchical structure from experts.
Step 3) Paired comparisons of the different criteria created in step 2 are organized into a square matrix called the paired comparison matrix. The basis of AHP is the comparison

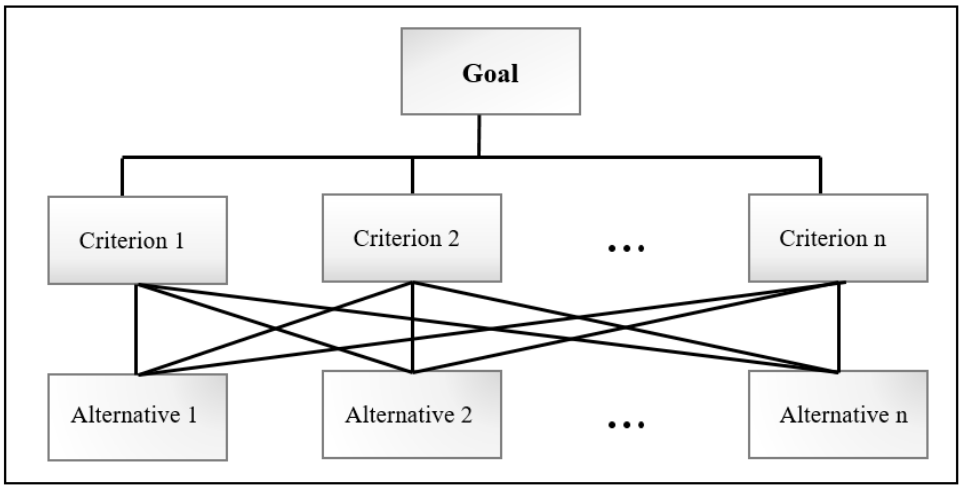

Figure 6: AHP hierarchical structure.

Table 2: The Saaty Rating Scale.

Intensity of Importance	**Definition**	**Explanation**
1	Equal preference	Equal contribution of two activities to the goal
3	Moderate preference	One activity is slightly more important than another
5	Strong preference	One activity is strongly more important than another
7	Very strong preference	One activity is very strongly more important than another
9	Extreme preference	One activity is extremely more important than another and is at the highest possible level of approval
2, 4, 6, 8	Intermediate values	When compromise is required
Reciprocals of the above	If there is one of the above non-zero numbers for activity i compared to activity j, j has a reciprocal value compared to i	

matrix that can be represented as Eq. (7) [14]:

$$C = \begin{cases} c_{ij} = \frac{1}{c_{ji}} & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{1}$$

Where $C \in \mathbb{R}^{k \times k}$.

Step 4) For each comparison matrix, a priority vector (vector of weights) must be calculated. Calculating the priority vector is one of the basic steps in AHP. There are several methods for computed the priority vector [13].

Step 5) The consistency of the comparison matrices is computed. If this consistency ratio fails to reach the required level, comparisons may be reconsidered. The Consistency Ratio (CR) is computed as follows [14]:

$$CR = \frac{CI}{RI} \tag{2}$$

Where RI represents the random index that can be computed randomly based on the rank of the comparison matrix. Some RI values are shown in Table 5. Also, CI represents the consistency index and is computed based on Eq. (9) [14]:

$$CI = \frac{\lambda_{max} - k}{k - 1} \tag{3}$$

Where λ_{max} indicates the maximum eigenvalue of the comparison matrix. Saaty has proved that the comparison matrix is consistent if $CR < 0.1$.

Table 3: Random Index (RI) [56].

n	1	2	3	4	5	6	7	8	9
RI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.51

4.2 Harris hawks optimization (HHO)

HHO is a recent gradient-free, population-based, and nature-inspired optimizer introduced by Heidari et al. [17] to solve global optimization problems. The HHO is a rapid, powerful, and high-performance optimization algorithm. The main idea of the HHO algorithm is to imitate the social behavior of Harris hawk in nature. The HHO algorithm is largely inspired by the chasing strategy and cooperative behavior of Harris hawks. In HHO, the prey is considered as the best solution (that is shown with the rabbit in the HHO). The stages of exploration and exploitation of the HHO optimizer are modeled by exploring a prey, performing a surprise pounce, and then attacking the target prey. Depending on the dynamic nature of the conditions and the escape behaviors of prey, the HHO algorithm can display various attack strategies. The logical and mathematical model of HHO consists of three main phases (i.e., exploration phase, the transition from exploration to exploitation, and the exploitation phase) which are represented in Fig. 9. These phases will be described in more detail below.

Exploration phase: This phase defines the hawks' position in exploring the prey. At this point, Harris Hawks searches for prey. In each iteration of HHO, the fitness value for each hawk is calculated based on the target prey because all hawks are candidate solutions. Although hawks can track and identify prey with their strong eyes, it is sometimes hard to see prey. Hence, the hawks are waiting and monitoring the site in the hope of seeing prey. Therefore, hawks identify prey based on two strategies. In the first strategy, hawks find the prey based on the position of the other members. In the second strategy, hawks detect the prey based on the perch on a random tree (X_{rand}).

$$X_i(t+1) = \begin{cases} X_{rand}(t) - r_1 \left| X_{rand}(t) - 2r_2 X(t) \right| & \text{if } \ q \geq 0.5 \\ (X_{prey}(t) - X_m(t)) - \omega & \text{if } \ q < 0.5 \end{cases} \tag{4}$$

Where $X_i(t+1)$ indicates the updated position of hawks in next iteration t, $X_{rand}(t)$ indicates the current position of hawks, r_1, r_2, r_3, r_4, and q indicates random numbers in the interval $(0, 1)$, $X_{prey}(t)$ indicates the position of prey, and $X_m(t)$ indicates the average of the positions for all hawks, which is computed according to Eq. (11):

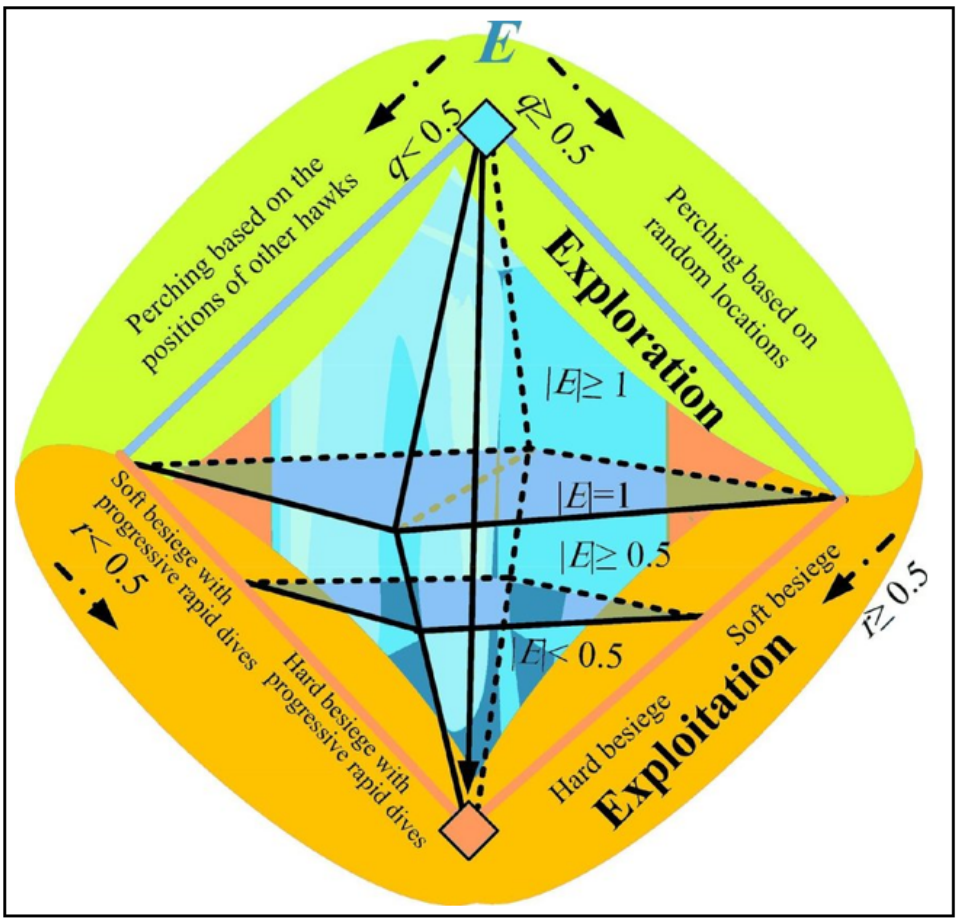

Figure 7: Various phases of HHO [17].

$$X_m(t) = \frac{\sum_{i=1}^{N} X_i(t)}{N} \tag{5}$$

$\omega = r_3(LB + r_4(UBLB))$ indicates the difference among upper and lower bounds of variables.

The transition from exploration to exploitation: At this phase, the hawks transfer from exploration to exploitation based on the escaping energy (E) of the prey. During the escape, the prey's energy decreases. The prey's energy can be modeled as below:

$$E = 2E_0\left(1 - \frac{t}{t_{max}}\right) \tag{6}$$

Where E represents escaping energy, E_0 indicates the initial state of energy that randomly changing in the interval $(-1, 1)$, and t_{max} indicates the total number of iterations. When the escaping energy of the prey $|E| \geq 1$, HHO guides the hawk to explore various areas

in search of the prey (exploration phase). When escaping energy of the prey is decreased $|E| < 1$, hawks are in the exploitation phase and they are searching the neighborhood to find a solution.
Exploitation Phase: At this phase, the Harris hawks attack the prey according to the situation discovered in the previous phase. The exploitation phase has two main elements: hawks chase strategies and prey escape behaviors because the prey always tries to escape and the hawks follow the chase strategy. To model the hawk's surprise pounce behavior on prey, the following chasing strategies are introduced:
i. Soft besiege,
ii. Hard besiege,
iii. Soft besiege with progressive rapid dives, and
iv. Hard besiege with progressive rapid dives.
In HHO, the two variables $|E|$ and r determine which strategy to use. $|E|$ represents the escaping energy of the prey and r represents the probability of escaping, if $r < 0.5$ indicates the prey has a higher chance of successful escape and if $r \geq 0.5$ indicates the failure to escape. The different types of exploitation and exploration phases are presented in Fig. 10.

5 Background

This section describes the background of the AHP and the HHO algorithm.

5.1 Analytic Hierarchy Process (AHP)

AHP is a multi-criteria decision-making tool for dealing with complex decision problems [7], [36]. In the AHP method, it is not necessary to define a complex expert system and AHP decides based on the set of evaluation criteria and a set of alternative options. The hierarchical model includes three levels of goal, criteria, and alternatives [10]. Figure 8 shows the hierarchical structure of AHP.
The relative values of alternatives or criteria are determined by the Saaty Rating Scale [41]. Table 4 shows the AHP numerical scales, which vary from 1 to 9.

Table 4: The Saaty Rating Scale.

Intensity of Importance	**Definition**	**Explanation**
1	Equal preference	Equal contribution of two activities to the goal
3	Moderate preference	One activity is slightly more important than another
5	Strong preference	One activity is strongly more important than another
7	Very strong preference	One activity is very strongly more important than another
9	Extreme preference	One activity is extremely more important than another and is at the highest possible level of approval
2, 4, 6, 8	Intermediate values	When compromise is required
Reciprocals of the above	If there is one of the above non-zero numbers for activity i compared to activity j, j has a reciprocal value compared to i	

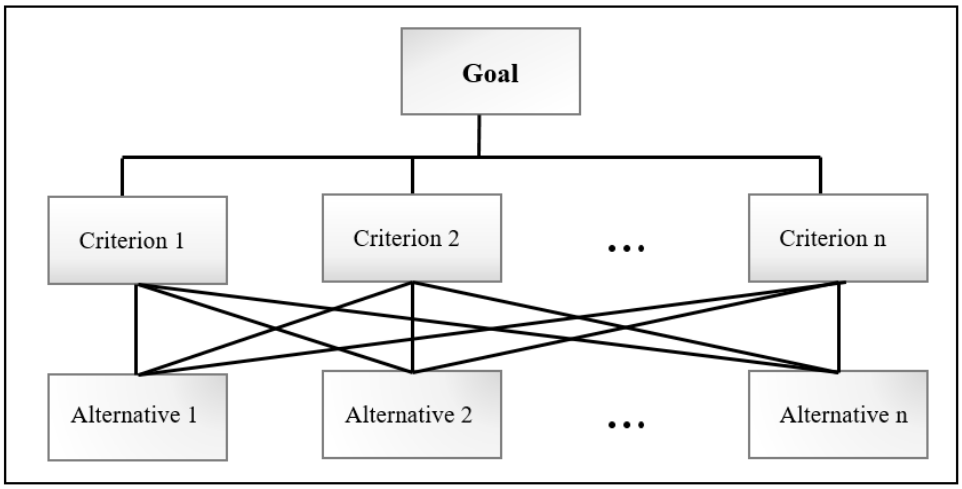

Figure 8: AHP hierarchical structure.

AHP provides a way to break down a problem into a hierarchy of sub-problems for easy evaluation. AHP steps are as follows:
Step 1) Divide the problem into hierarchies of goal, criteria, and alternatives.
Step 2) Collect data of the hierarchical structure from experts.
Step 3) Paired comparisons of the different criteria created in step 2 are organized into a square matrix called the paired comparison matrix. The basis of AHP is the comparison matrix that can be represented as Eq. (7) [14]:

$$C = \begin{cases} c_{ij} = \frac{1}{c_{ji}} & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{7}$$

Where $C \in \mathbb{R}^{k \times k}$.
Step 4) For each comparison matrix, a priority vector (vector of weights) must be calculated. Calculating the priority vector is one of the basic steps in AHP. There are several methods for computed the priority vector [13].
Step 5) The consistency of the comparison matrices is computed. If this consistency ratio fails to reach the required level, comparisons may be reconsidered. The Consistency Ratio (CR) is computed as follows [14]:

$$CR = \frac{CI}{RI} \tag{8}$$

Where RI represents the random index that can be computed randomly based on the rank of the comparison matrix. Some RI values are shown in Table 5. Also, CI represents the consistency index and is computed based on Eq. (9) [14]:

$$CI = \frac{\lambda_{max} - k}{k - 1} \tag{9}$$

Where λ_{max} indicates the maximum eigenvalue of the comparison matrix. Saaty has proved that the comparison matrix is consistent if $CR < 0.1$.

Table 5: Random Index (RI) [56].

n	1	2	3	4	5	6	7	8	9
RI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.51

5.2 Harris hawks optimization (HHO)

HHO is a recent gradient-free, population-based, and nature-inspired optimizer introduced by Heidari et al. [17] to solve global optimization problems. The HHO is a rapid, powerful, and high-performance optimization algorithm. The main idea of the HHO algorithm is to imitate the social behavior of Harris hawk in nature. The HHO algorithm is largely inspired by the chasing strategy and cooperative behavior of Harris hawks. In HHO, the prey is considered as the best solution (that is shown with the rabbit in the HHO). The stages of exploration and exploitation of the HHO optimizer are modeled by exploring a prey, performing a surprise pounce, and then attacking the target prey. Depending on the dynamic nature of the conditions and the escape behaviors of prey, the HHO algorithm can display various attack strategies. The logical and mathematical model of HHO consists of three main phases (i.e., exploration phase, the transition from exploration to exploitation, and the exploitation phase) which are represented in Fig. 9. These phases will be described in more detail below.

Exploration phase: This phase defines the hawks' position in exploring the prey. At this point, Harris Hawks searches for prey. In each iteration of HHO, the fitness value for each hawk is calculated based on the target prey because all hawks are candidate solutions. Although hawks can track and identify prey with their strong eyes, it is sometimes hard to see prey. Hence, the hawks are waiting and monitoring the site in the hope of seeing prey. Therefore, hawks identify prey based on two strategies. In the first strategy, hawks find the prey based on the position of the other members. In the second strategy, hawks detect the prey based on the perch on a random tree (X_{rand}).

$$X_i(t+1) = \begin{cases} X_{rand}(t) - r_1 \left| X_{rand}(t) - 2r_2 X(t) \right| & \text{if } \ q \geq 0.5 \\ (X_{prey}(t) - X_m(t)) - \omega & \text{if } \ q < 0.5 \end{cases} \tag{10}$$

Where $X_i(t+1)$ indicates the updated position of hawks in next iteration t, $X_{rand}(t)$ indicates the current position of hawks, r_1, r_2, r_3, r_4, and q indicates random numbers in the interval $(0, 1)$, $X_{prey}(t)$ indicates the position of prey, and $X_m(t)$ indicates the average of the positions for all hawks, which is computed according to Eq. (11):

$$X_m(t) = \frac{\sum_{i=1}^{N} X_i(t)}{N} \tag{11}$$

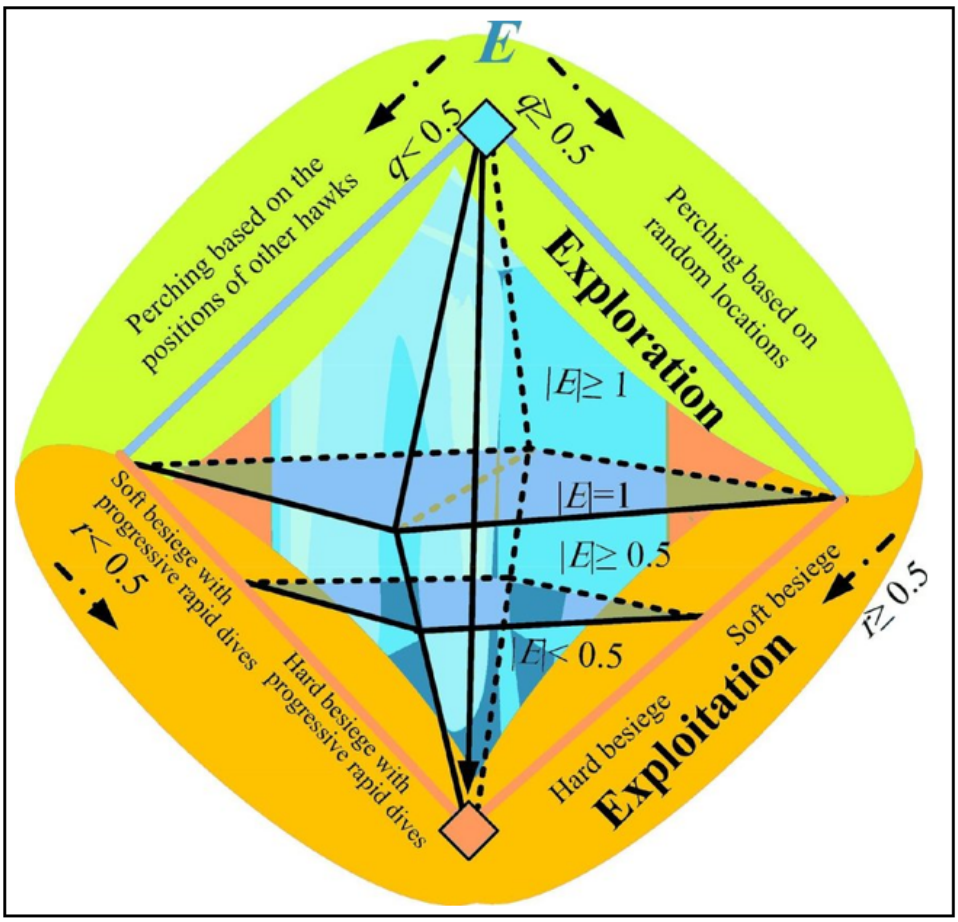

Figure 9: Various phases of HHO [17].

$\omega = r_3(LB + r_4(UBLB))$ indicates the difference among upper and lower bounds of variables.

The transition from exploration to exploitation: At this phase, the hawks transfer from exploration to exploitation based on the escaping energy (E) of the prey. During the escape, the prey's energy decreases. The prey's energy can be modeled as below:

$$E = 2E_0\left(1 - \frac{t}{t_{max}}\right) \tag{12}$$

Where E represents escaping energy, E_0 indicates the initial state of energy that randomly changing in the interval $(-1, 1)$, and t_{max} indicates the total number of iterations. When the escaping energy of the prey $|E| \geq 1$, HHO guides the hawk to explore various areas in search of the prey (exploration phase). When escaping energy of the prey is decreased $|E| < 1$, hawks are in the exploitation phase and they are searching the neighborhood to find a solution.

Exploitation Phase: At this phase, the Harris hawks attack the prey according to the situation discovered in the previous phase. The exploitation phase has two main elements:

hawks chase strategies and prey escape behaviors because the prey always tries to escape and the hawks follow the chase strategy. To model the hawk's surprise pounce behavior on prey, the following chasing strategies are introduced:
i. Soft besiege,
ii. Hard besiege,
iii. Soft besiege with progressive rapid dives, and
iv. Hard besiege with progressive rapid dives.
In HHO, the two variables $|E|$ and r determine which strategy to use. $|E|$ represents the escaping energy of the prey and r represents the probability of escaping, if $r < 0.5$ indicates the prey has a higher chance of successful escape and if $r \geq 0.5$ indicates the failure to escape. The different types of exploitation and exploration phases are presented in Fig. 10.

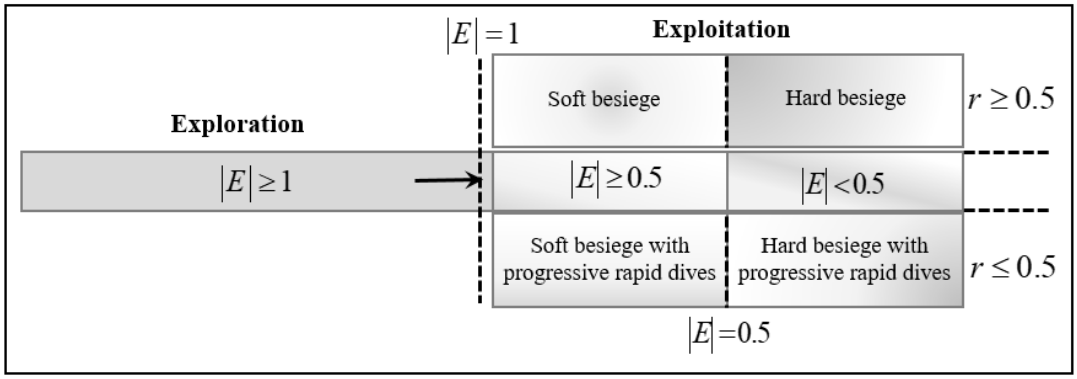

Figure 10: The exploration phase and the exploitation phases of the HHO [47].

i) Soft besiege: It $|E| \geq 0.5$ and $r \geq 0.5$, then soft besiege occurs. Since in this case the prey still has some energy to escape, the hawks softly surround the prey to losing more energy before a surprise pounce. Thus, the prey cannot escape successfully because the prey's energy is discharged during the escape from the hawks. The soft besiege is modeled by Eq. (13).

$$X_i(t+1) = \Delta X(t) - E\,|JX_{prey}(t) - X(t)| \tag{13}$$

$$\Delta X(t) = X_{prey}(t) - X(t) \tag{14}$$

Where $\Delta X(t)$ represents the difference between prey position and the current position in iteration t, $J = 2\,(1 - r_5)$ refers to the random jump strength of the prey throughout the escaping procedure which changed randomly in each iteration, and r_5 refers to a random number in the interval $(0, 1)$.
ii) Hard besiege: When $|E| < 0.5$ and $r \geq 0.5$ hard besiege takes place. In this case, the prey cannot escape successfully because it is very tired and has a low chance of escape. In this position, the hawk hardly encircles around the prey to make the final surprise pounce. The updated positions of the hawks are obtained by Eq. (15).

$$X(t+1) = X_{prey}(t) - E\,|\Delta X(t)| \tag{15}$$

iii) Soft besiege with progressive rapid dives: If $|E| \geq 0.5$ and $r < 0.5$, then soft besiege with progressive rapid dives occurs. In this case, the prey still has enough energy to successfully escape from the attack. In such a situation, the hawks must decide to make the best possible dive toward the prey. For this purpose, the hawks can make several moves, evaluate the new moves using Eq. (16), and then compare the result of the move with the last dive towards the prey. If the result of the comparison does not lead to the specification of the best dive towards the prey, then hawks start performing rapid and irregular dives based on Levy flight (LF), as formulated in Eq. (17).

$$Y = X_{prey}(t) - E|JX_{prey}(t) - X(t)| \tag{16}$$

$$Z = Y + S \times LF(D) \tag{17}$$

Where D and S show the problem dimension and a random vector with size $1 \times D$, respectively. The levy flight function (LF) is calculated as below:

$$LF = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}},\ \sigma = \left(\frac{\Gamma\left(1+\beta\right) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\frac{(\beta-1)}{2}}} \right)^{\frac{1}{\beta}} \tag{18}$$

Where u and v indicate random values in the range $(0, 1)$, and β indicates a constant set to 1.5.

Thus, the final equation for updating the positions of hawks in soft besiege with progressive rapid dives strategy can be computed as follows:

$$X(t+1) = \begin{cases} Y & \text{if } F(Y) < F(X(t)) \\ Z & \text{if } F(Z) < F(X(t)) \end{cases} \tag{19}$$

Where F represents a fitness function for an optimization problem, and Y and Z are computed using Eq. 16 and Eq. 17, respectively.

iv) Hard besiege with progressive rapid dives: If $|E| < 0.5$ and $r < 0.5$, then hard besiege with progressive rapid dives takes place. In this case, the prey does not have enough energy to escape, and the hawks create a hard besiege. In such a situation, the Harris hawks try to approach the prey with rapid dives to reduce the distance before making a surprise pounce to catch the prey. This strategy is modeled based on Eq. 20.

$$X(t+1) = \begin{cases} Y' & \text{if } F(Y') < F(X(t)) \\ Z' & \text{if } F(Z') < F(X(t)) \end{cases} \tag{20}$$

Where Y' and Z' are calculated as follows:

$$Y' = X_{prey}(t) - E|JX_{prey}(t) - X_m(t)| \tag{21}$$

$$Z' = Y' + S \times LF(D) \tag{22}$$

Where $X_m(t)$ is obtained using Eq. 11.
The flowchart of the HHO algorithm is represented in Fig. 11.

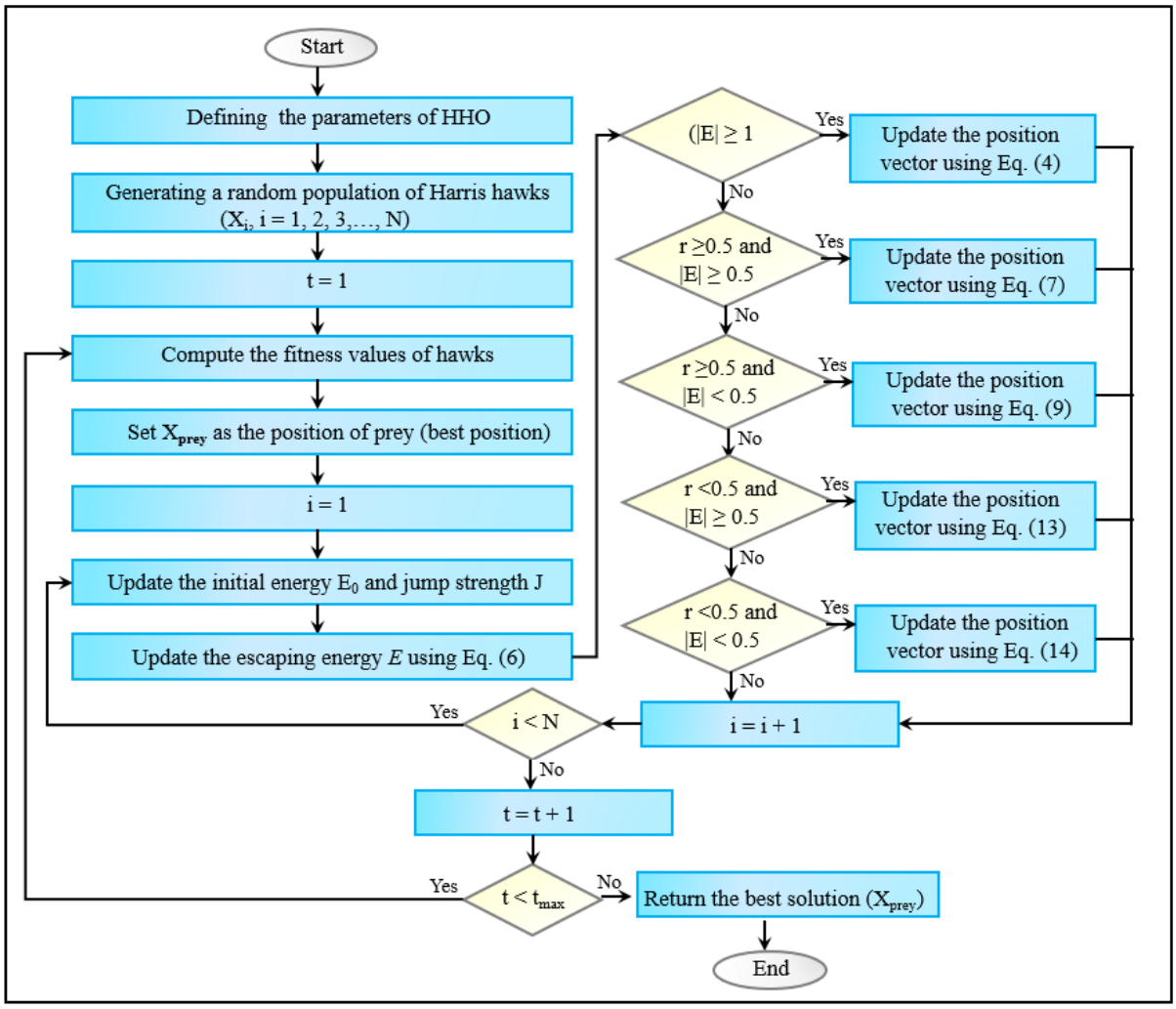

Figure 11: Flowchart of HHO algorithm.

6 Proposed Task Scheduling Algorithm (PHHO)

Figure 12 describes the proposed algorithm architecture in the cloud environment. The cloud environment receives a variety of tasks. The PHHO algorithm applies the AHP to manage the priority of tasks based on length and memory. The main goal of the proposed task algorithm is to decrease makespan and energy consumption while improving resource utilization and throughput. when tasks are prioritized, they are managed in the

task queue. The proposed algorithm uses HHO to optimally assign tasks to resources. This section includes four sub-sections. In subsection 6.1, the related concepts of task scheduling in the cloud system are given. Subsection 6.2 describes the hierarchical process for assigning priority to tasks, Subsection 6.3 describes the objective functions and mathematical models, and Subsection 6.4 presents the proposed algorithm.

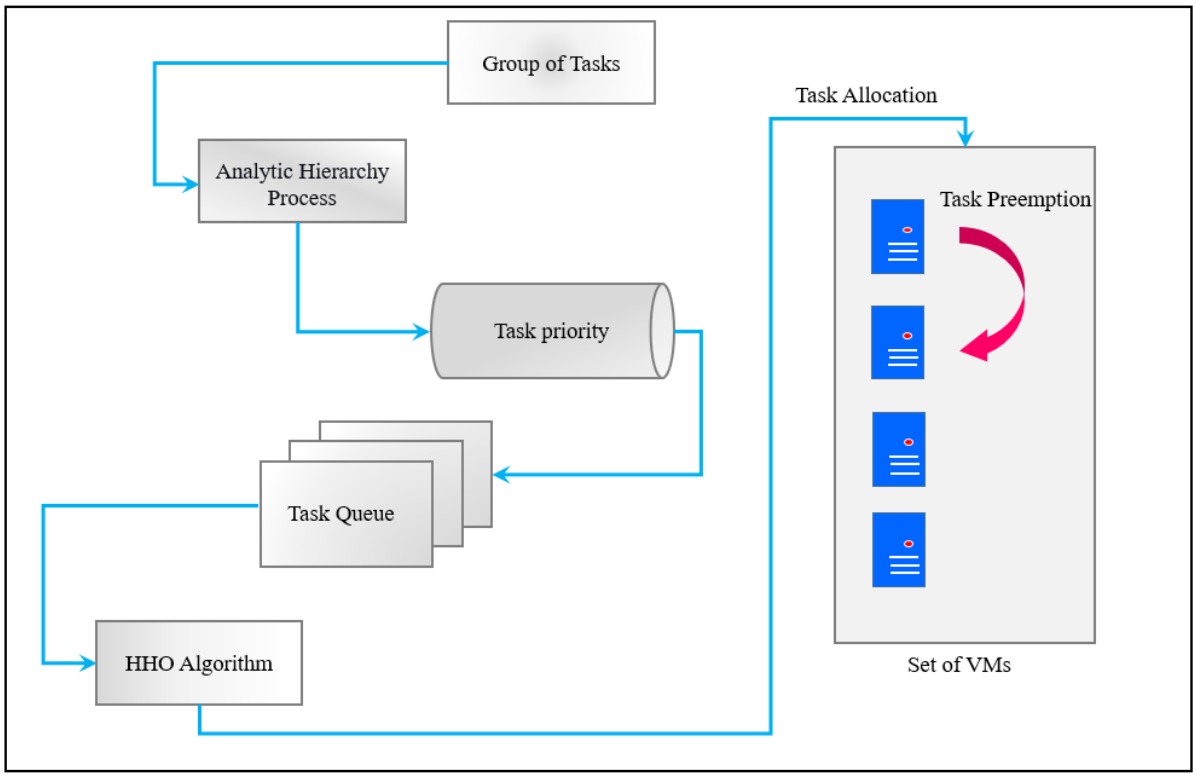

Figure 12: Presented architecture for task allocation.

6.1 System Model

Task Scheduling is the utilization of available resources by organizing incoming requests (tasks) in a specific style. In the cloud environment, tasks are scheduled to improve the performance of different QoS parameters. To model the task scheduling problem, it is assumed that all submitted tasks are independent, tasks cannot migrate between VMs, and VMs are heterogeneous and have various processing and power consumption capabilities. Consider the cloud consists of m number of VMs that are represented as $VM = \{VM_1, VM_2, ..., VM_m\}$, where VM_j indicates the $j - th$ VM. The task set is defined by $T = \{T_1, T_2, ..., T_n\}$, where T_i represents the $i - th$ task.

6.2 Analytic Hierarchy Process for task priorities

The hierarchical process optimizes decisions based on various criteria. The AHP as a multi-criteria decision-maker has different applications such as strategic scheduling, business or public policy, resource allocation, resource selection, and many more. One of AHP's applications is that it can be used to prioritize tasks in the task scheduling problem. The PHHO algorithm applies the AHP to prioritize input tasks. Before executing the optimization algorithm, a hierarchical process is applied to the tasks. The proposed algorithm considers two parameters (i.e. task length and task memory). Tasks priorities are determined by task length and task memory. In the PHHO algorithm, the arithmetic mean method is applied to calculate the weights. The AHP procedure for prioritizing tasks is shown in Algorithm 1.

Algorithm 1. AHP

Input: Set of tasks $T = \{T_1, T_2, ..., T_n\}$ includes their length $L = \{L_1, L_2, ..., L_n\}$ and memory $M = \{M_1, M_2, ..., M_n\}$
Output: Prioritized tasks T_P
Begin
1. Submit input T to the cloud
2. Consider the parameters length and memory of each task T_i
3. Assign priority to each task to create comparison matrices
4. Add the values on each column $X = (X_1, X_2, , X_n)$
5. Divide each matrix element into X_i
6. Convert the value of each element to a decimal
7. Calculate the average of each row
8. Calculate the final weight by the sum of the product of the significance of the criteria in the weight of the alternatives
9. Calculate Consistency Ratio using Eq. (8)
10. Rearrange tasks according to priorities
End

6.3 Objective Function

This section explains the different objectives (i.e., makespan, resource utilization, throughput, and energy consumption) that are used during the scheduling process. The objectives are makespan, resource utilization, throughput, and energy consumption. The objective function is computed as follows:
Makespan: Makespan can be defined as the time it takes from the moment a user sends a request to the completion of the last task. Minimizing the entire execution time, while mapping tasks to VMs, also decreases execution costs. Therefore, makespan is considered as one of the principal objectives in task scheduling, which decreases the length of the schedule while meeting the needs of the user. Makespan can be defined by Eq. (23) [22]:

$$MS = \max\left(ET_j\right),\ 1 \leq j \leq m \tag{23}$$

Where ET_j represents the execution time of the VM_j and it is computed based on the decision variable A_{ij}.

$$A_{ij} = \begin{cases} 1 & \text{if } T_i \textit{ is assigned to } VM_j \\ 0 & \text{if } T_i \textit{ is not assigned to } VM_j \end{cases} \tag{24}$$

$$ET_j = \sum_{i=1}^{n} A_{ij} \times CT_{ij} \tag{25}$$

Where CT_{ij} represents the completion time of the task T_i in the VM_j and it is calculated by Eq. (26):

$$CT_{ij} = \frac{Len_i}{PT_j} \tag{26}$$

Where Len_i is the length of the task T_i and length of the task is described in terms of the number of instructions (Millions of Instruction) and PT_j is the processing time of the VM_j in the cloud.

Resource Utilization: Resource utilization is another important parameter in task scheduling, which refers to the most use of leased cloud resources for effective task allocation. This parameter is very significant for CSPs because service providers want to get the maximum profit by renting a limited number of resources. Any time when leased resources (VMs) are not used is a cost. We try to maximize the utilization of resources because it decreases costs and increases the profits of service providers. The resources utilization with an inverse linear relationship is related to makespan as follows [35]:

$$RU(VM_{ij}) = \frac{CT_{ij}}{MS} \tag{27}$$

The average resource utilization can be calculated by Eq. (28):

$$ARU(VM_{ij}) = \frac{\sum_{j=1}^{m} RU(VM_{ij})}{m} \tag{28}$$

Throughput: Throughput is the rate of success at which a resource can execute a certain number of tasks in a given time. In other words, it measures the number of tasks completed per unit of time. The term throughput is commonly applied to show the ability of a system to deliver (i.e., the speed rate in processing) customer requests. It can be computed by Eq. (29) [35]:

$$Th = \frac{RU(VM_{ij})}{MS} \tag{29}$$

The average throughput can be applied by Eq. (30).

$$ATh = \sum_{j=1}^{m} Th \tag{30}$$

Energy Consumption: In recent years, reducing energy consumption has become one of the critical challenges of organizations, and governments. There are worldwide concerns about minimizing energy consumption, as increasing energy consumption will increase carbon emissions and hurt the environment. In the cloud environment, executing tasks

consumes a lot of energy due to a large number of tasks. During the execution of tasks, a VM can be in an active state or idle state. The idle state of a VM consumes 60%-70% of the energy consumption of the active state of that VM. Accordingly, consider α_j joules/ Millions of instruction consumed by VM_j in the active state and β_j joules/Millions of instruction consumed by VM_j in the idle state. A VM_j remains (ET_j) second in active state and $(MS - ET_j)$ second in the idle state. The mathematical representation of total energy consumption can be calculated by Eq. (31) [4]:

$$TEC = \sum_{j=1}^{m} [[ET_j \times \alpha_j + (MS - ET_j) \times \beta_j] \times PS_j] \tag{31}$$

To measure the efficiency of each algorithm and user satisfaction, the makespan must be minimized. Resource utilization must be considerably improved on the server to maximize overall system throughput. Energy must also be minimized to increase system efficiency. Although these parameters are conflicting, the PHHO algorithm has considered all four parameters in the fitness function (e.g., although using a powerful CPU speeds up the processing of a task, energy consumption also increases). The effectiveness of each meta-algorithm is evaluated based on a fitness function that is problem-dependent. Therefore, the fitness function is formulated by considering the weight value for each objective as follows:

$$F_{opt} = \min \left\{ \lambda_1 \times \frac{MS}{MaxMS} + \lambda_2 \times \frac{1}{ARU} + \lambda_3 \times \frac{1}{ATh} + \lambda_4 \times \frac{TEC}{MaxEC} \right\} \tag{32}$$

Where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are weight values in the range $[0, 1]$, $MaxMS$ indicates maximum makespan, and $MaxEC$ indicates maximum energy consumption.

6.4 Proposed Algorithm

The flowchart of the PHHO algorithm is demonstrated in Fig. 13. The major steps of the presented algorithm can be summarized as below:
1) Initialize parameters such as number of tasks, number of VMs, upper bound and lower bound, number of search agents, positions of Harris hawks, and maximum number of iterations.
2) Prioritize tasks according to the AHP model.
3) Rearrange tasks based on their priority.
4) Start searching for the optimal solution using the HHO algorithm. In this step, according to the position of each hawk, the fitness value of each hawk is calculated using Eq. (32). The hawk with the smallest fitness value is recorded as the current optimal solution (considered as the position of the prey).
5) Update the position of the Harris hawks.
6) When the position of all the hawks is updated, one iteration is performed. If the maximum number of iterations is reached, the search process ends. Otherwise, go to step 4 for a new search. Once the specified iterations are reached, the prey position is

considered the best solution and transferred to the decision variables a_{ij} as the best task scheduling solution.

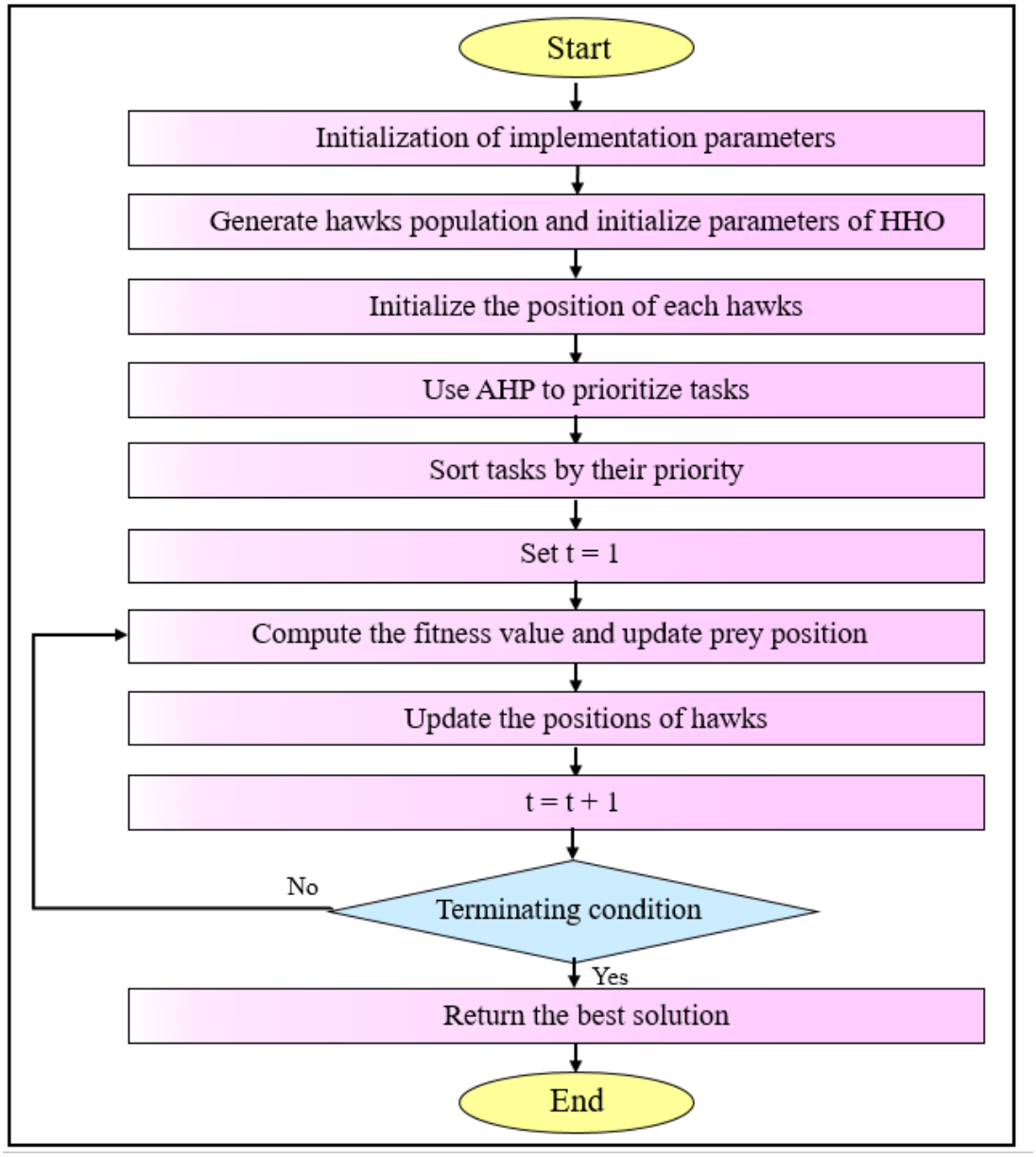

Figure 13: Flowchart of the PHHO algorithm.

7 Experimental results and performance analysis

The scheduling algorithms are simulated on Intel(R) Core(TM) i5-7200U CPU with 2.50 GHz, Windows 10 Pro platform, and using MATLAB R2018a. The meta-heuristic algo-

rithms proposed for comparison are GA [18], PSO [23], ACO [9], MFO [31], WOA [33], and SSA [32]. To evaluate the efficiency of the PHHO algorithm, the comparison is based on four parameters: makespan, resource utilization, throughput, and energy consumption. In the following, four different scenarios are presented to validate the proposed algorithm.

7.1 Simulation with different numbers of tasks

In this scenario, the number of input tasks varies from 200 to 800 at intervals of 200, and the total number of VMs is considered constant. Table 6 lists the characteristics of the cloud system and the HHO algorithm.

Table 6: Simulation environment (different number of tasks).

Parameters	**Values**
Number of tasks	200-800
Tasks size (MI)	100-5000
Task Memory	256-512 MB
Number of VMs	50
VMs execution speed (MIPS)	500-9000
Maximum iteration	100
Population size	50
E_0	(-1, 1)

Makespan represents the necessary time to complete all tasks of the system. Measuring the makespan is one of the most significant parameters in the scheduling problem because reducing the makespan helps to minimize execution cost and meet the deadline of the task. Figure 14 represents the performance result of the PHHO algorithm for makespan. The y-axis shows the effect on a makespan when the number of tasks is increasing. As illustrated in Fig. 14, with the increasing number of tasks, makespan increases. In addition, the HHO algorithm performs better in terms of makespan minimization compared to other methods. Makespan minimization by HHO is 27%38% lower than that of GA for 200 through 800 number of tasks, respectively. Also, makespan minimization by HHO is 74%49% lower than that of ACO for 200 through 800 number of tasks, respectively. This is because GA and ACO have poor exploitation capability.

The resource utilization criterion represents how resources are used. It can be observed from Fig. 15 that the HHO algorithm is the best in terms of resource utilization for a higher number of tasks. The comparison results for 400 tasks show that calculated resource utilization by HHO is approximately 29%, 20%, 58%, 16%, 12%, 14%, more than GA, PSO, ACO, WOA, SSA, and MFO, respectively.

Higher throughput values indicate that the scheduling algorithm is more efficient. High throughput means that more tasks are executed per unit time. Figure 16 represents the experimental results for different algorithms. It can be observed that HHO for 800 tasks achieves 59%, 27%, 71%, 19%, 13%, and 24% higher throughput than GA, PSO, ACO, WOA, SSA, and MFO, respectively. This indicates that HHO has increased scheduling efficiency in terms of the number of tasks processed by VMs.

In the cloud data center, task scheduling is the heart of successful energy management. CPU utilization and resource utilization will directly affect a task's energy consumption.

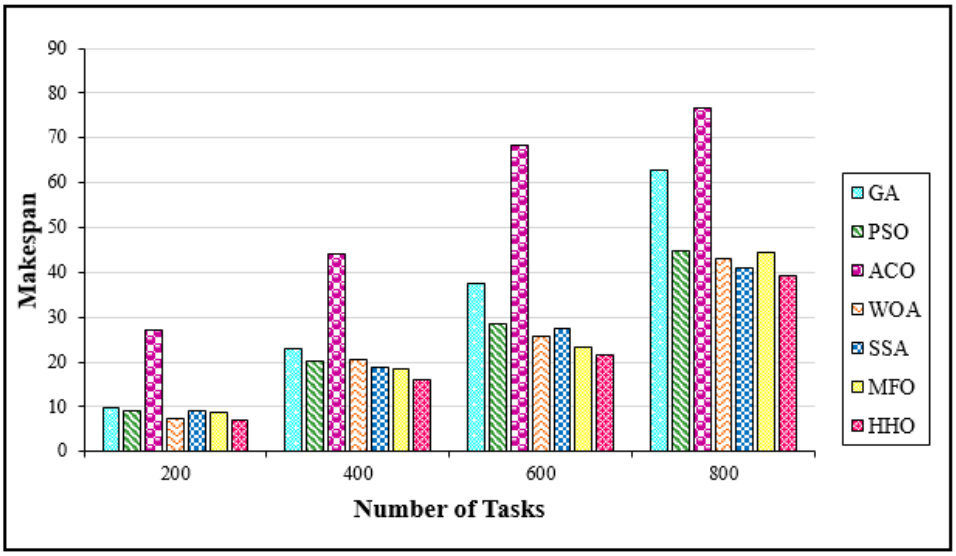

Figure 14: Makespan for the different number of tasks.

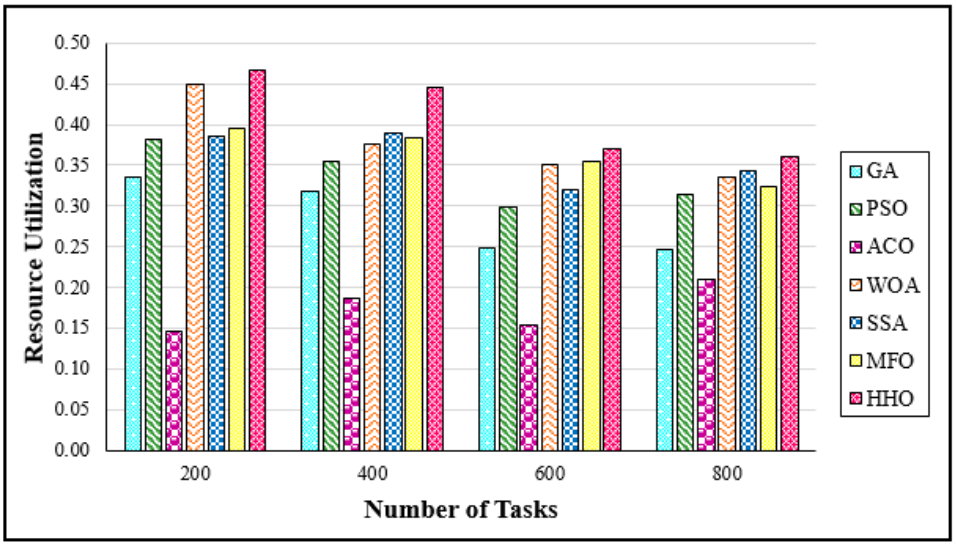

Figure 15: Resource utilization for the different number of tasks.

If the CPU is not used properly, the energy consumption will be high because idle power has not been used effectively. Sometimes due to high demand for resources, energy consumption is high and this may reduce the efficiency of the system. Scheduling decisions are significant to finding the right assignment of tasks to resources to reduce energy consumption by resources. Figure 17 shows the amount of energy consumption using different algorithms. As the results show, HHO reduces energy consumption compared to other algorithms and thus improves performance. Also, it can be seen that ACO has the worst performance among all algorithms due to its low exploitation capability. The results in-

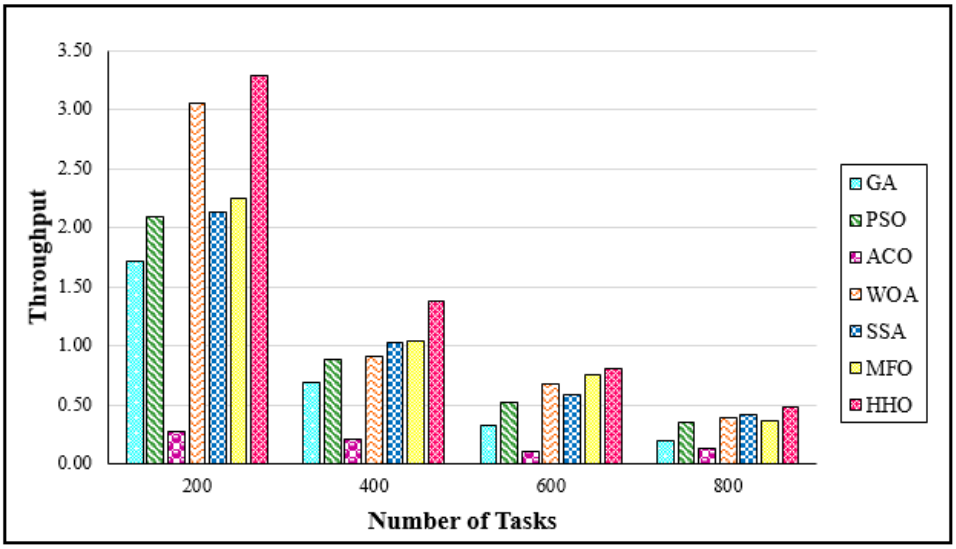

Figure 16: Throughput for the different number of tasks.

dicate that HHO can be used effectively for task scheduling for the different number of tasks with a fixed number of VMs. This is because HHO does not get involved in local optimal and has good exploration and exploitation capabilities.

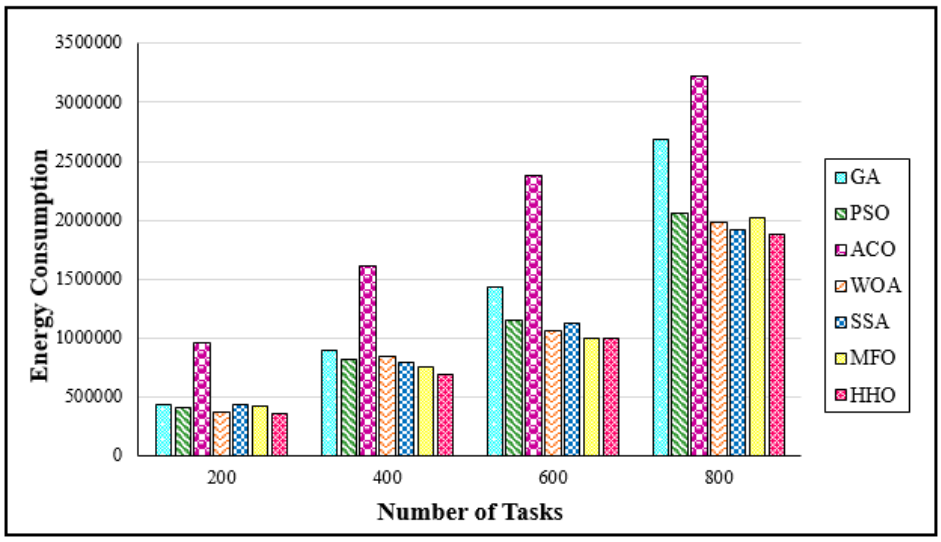

Figure 17: Energy consumption for the different number of tasks.

7.2 Simulation with different numbers of VMs

In the second scenario, the number of input tasks is 500 which is fixed, and the number of VMs varies from 10 to 40 at intervals of 10. The parameters setting are shown in Table

7.

Table 7: Simulation environment (different number of VMs).

Parameters	**Values**
Number of tasks	500
Tasks size (MI)	100-5000
Task Memory	256-512 MB
Number of VMs	10-40
VMs execution speed (MIPS)	500-9000
Maximum iteration	100
Population size	60
E_0	(-1, 1)

To have efficient scheduling, makespan must be decreased. Makespan is described as the maximum completion time of the tasks between VMs. Figure 23 represents the performance of different algorithms in terms of makespan for various numbers of VMs. It is expected that the makespan reduces as the number of VMs increases. These results show that HHO improves the performance compared to other algorithms and the use of a variable number of VMs does not negatively affect HHO performance. The improvement rates for 40 VMs are 42%, 21%, 67%, 15%, 38%, and 31% over GA, PSO, ACO, WOA, SSA, and MFO, respectively.

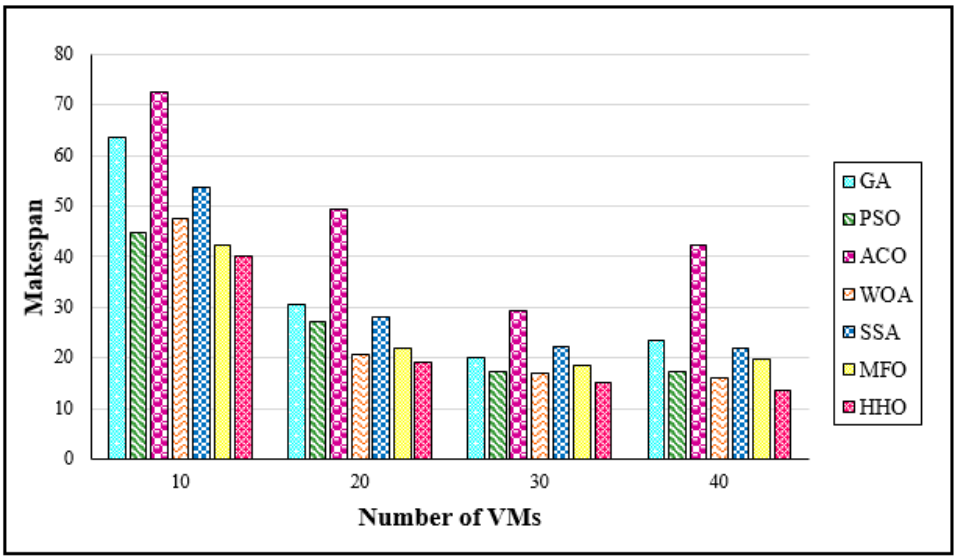

Figure 18: Makespan for the different number of VMs.

One of the main parameters in scheduling is maximizing resource utilization, which means keeping resources as busy as possible. An efficient scheduling algorithm is needed to make better utilization of resources. Resource utilization is significantly affected by the makespan reduction for different numbers of VMs. This is due to the consideration of fitness function which is inversely proportional to the makespan. Figure 24 represents the improvement of the HHO algorithm in resource utilization compared to the other

algorithms with different numbers of VMs. Also, it can be found that ACO has the worst performance compared to other algorithms. The X-axis represents the number of VMs, and the Y-axis indicates resource utilization.

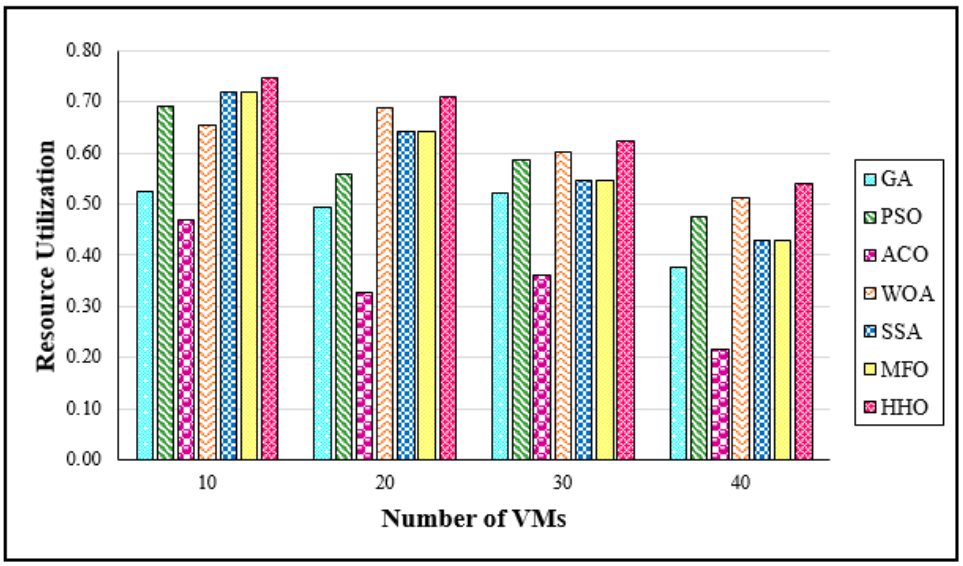

Figure 19: Resource utilization for the different number of VMs.

Throughput is a significant criterion in specifying the success of an efficient scheduling algorithm. Throughput is the number of tasks being processed by VMs per unit time. Higher values of throughput indicate better results. The results are illustrated in Figure 25. It can be observed that the throughput maximization by HHO is 40%52% higher than that of SSA for 10 through 40 VMs, respectively. Also, the throughput maximization by HHO is 8%40% higher than that of MFO for 10 through 40 VMs, respectively.
Energy consumption is one of the key parameters in maximizing the overall performance of the cloud system. There is a direct relationship between energy consumption and resource utilization because the optimal use of resources reduces energy consumption in a server. Figure 19 shows the energy consumption of varying VMs for different algorithms. As shown in Fig. 26, the energy consumption is proportional to the number of VMs in the cloud. As the number of VMs increases, the energy consumption increases. The proposed algorithm decreases energy consumption by up to 29%, 11%, 57%, 6%, 27%, and 18%, compared to GA, PSO, ACO, WOA, SSA, and MFO, respectively. It can be seen that the HHO can perform better for a various number of VMs with a certain number of tasks than other algorithms. This is due to the balance between the exploration and exploitation capability of the HHO algorithm.

7.3 Simulation with different number of iterations

In this scenario, the number of input tasks and the number of VMs are fixed (see Table 10).

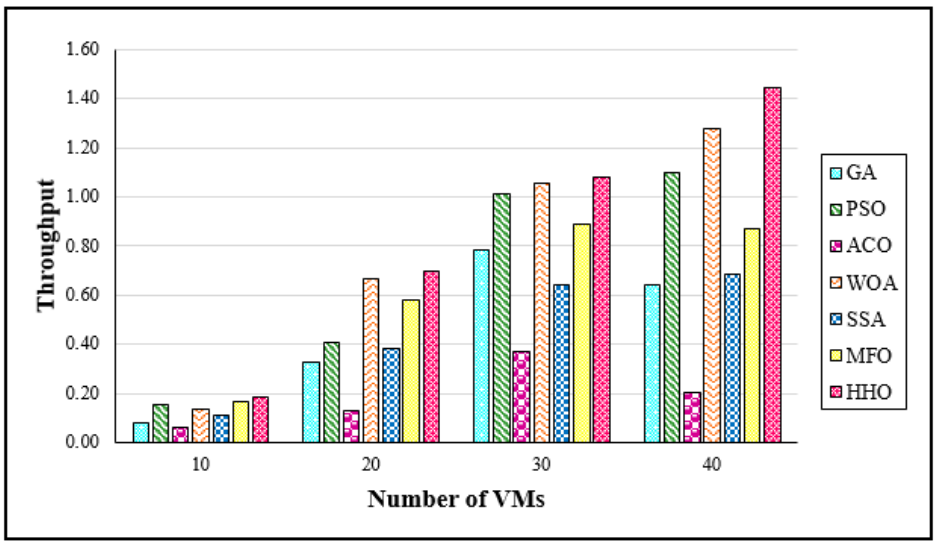

Figure 20: Throughput for the different number of VMs.

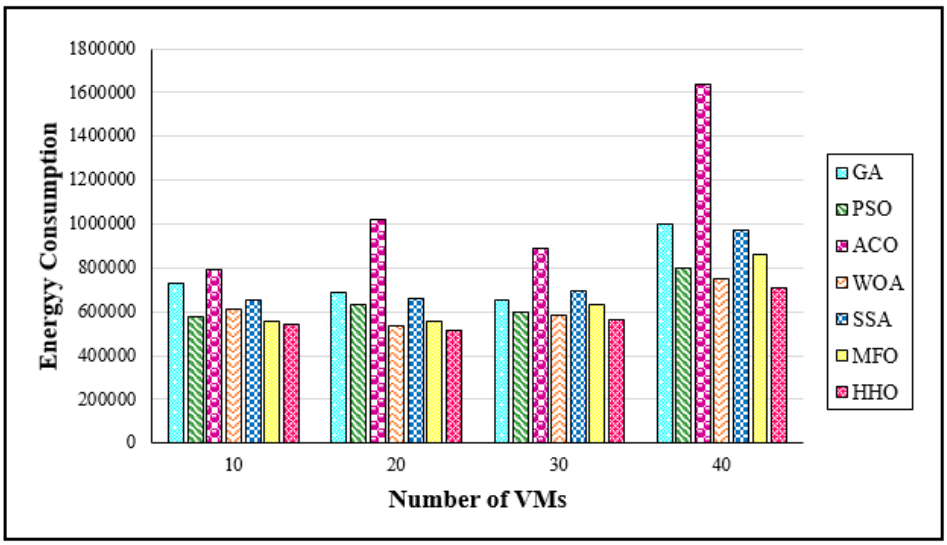

Figure 21: Energy consumption for the different number of VMs.

Figure 27 shows a comparison of seven algorithms with an increasing number of iterations. As can be seen, PSO and WOA work better than GA and ACO, respectively. Because GA and ACO have poor exploitation capability. Also, it can be found that SSA and MFO are more efficient than PSO and WOA because they have better search capability. In addition, it can be found that HHO has the best performance. This is because HHO can find its optimal solution faster than other algorithms. In other words, HHO has much better convergence speed and accuracy than other algorithms. Therefore, it can be found that HHO can have a strong ability to solve complex optimization problems due to its

Table 8: Simulation environment (different number of iterations).

Parameters	Values
Number of tasks	500
Tasks size (MI)	100-5000
Task Memory	256-512 MB
Number of VMs	25
VMs execution speed (MIPS)	500-9000
Maximum iteration	100
Population size	60
E_0	(-1, 1)

suitable transition from exploration to exploitation.

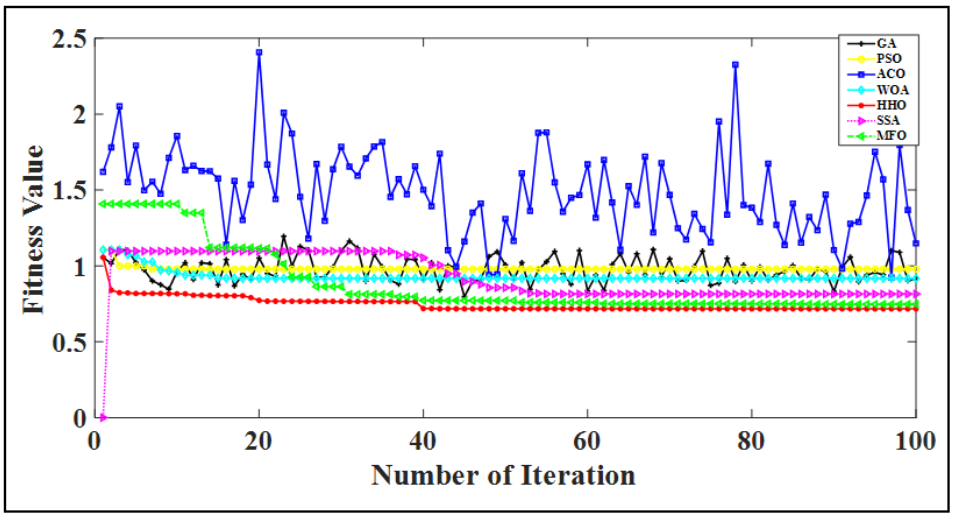

Figure 22: Convergence plot according to the number of iterations.

7.4 Comparison between priority and non-priority scheduling algorithm

In the last scenario, the proposed algorithm is compared with the case that does not take into account the priority. Table 11 shows the setting of parameters.

Table 9: Simulation environment (different number of iterations).

Parameters	Values
Number of tasks	800
Tasks size (MI)	100-5000
Task Memory	256-512 MB
Number of VMs	15
VMs execution speed (MIPS)	500-9000
Maximum iteration	100
Population size	40
E_0	(-1, 1)

To have efficient scheduling, makespan must be decreased. Makespan is described as the maximum completion time of the tasks between VMs. Figure 23 represents the performance of different algorithms in terms of makespan for various numbers of VMs. It is expected that the makespan reduces as the number of VMs increases. These results show that HHO improves the performance compared to other algorithms and the use of a variable number of VMs does not negatively affect HHO performance. The improvement rates for 40 VMs are 42%, 21%, 67%, 15%, 38%, and 31% over GA, PSO, ACO, WOA, SSA, and MFO, respectively.

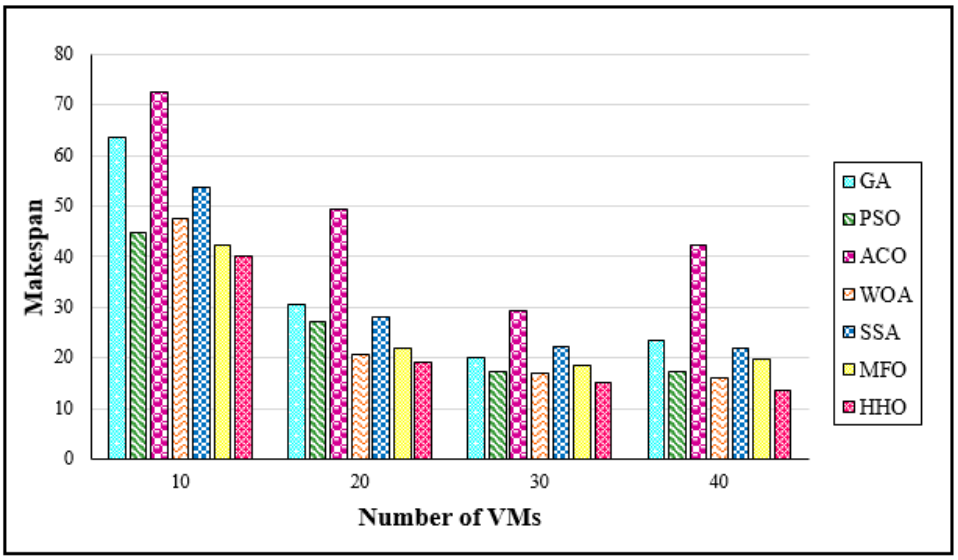

Figure 23: Makespan for the different number of VMs.

One of the main parameters in scheduling is maximizing resource utilization, which means keeping resources as busy as possible. An efficient scheduling algorithm is needed to make better utilization of resources. Resource utilization is significantly affected by the makespan reduction for different numbers of VMs. This is due to the consideration of fitness function which is inversely proportional to the makespan. Figure 24 represents the improvement of the HHO algorithm in resource utilization compared to the other algorithms with different numbers of VMs. Also, it can be found that ACO has the worst performance compared to other algorithms. The X-axis represents the number of VMs, and the Y-axis indicates resource utilization.

Throughput is a significant criterion in specifying the success of an efficient scheduling algorithm. Throughput is the number of tasks being processed by VMs per unit time. Higher values of throughput indicate better results. The results are illustrated in Figure 25. It can be observed that the throughput maximization by HHO is 40%52% higher than that of SSA for 10 through 40 VMs, respectively. Also, the throughput maximization by HHO is 8%40% higher than that of MFO for 10 through 40 VMs, respectively.

Energy consumption is one of the key parameters in maximizing the overall performance of the cloud system. There is a direct relationship between energy consumption and

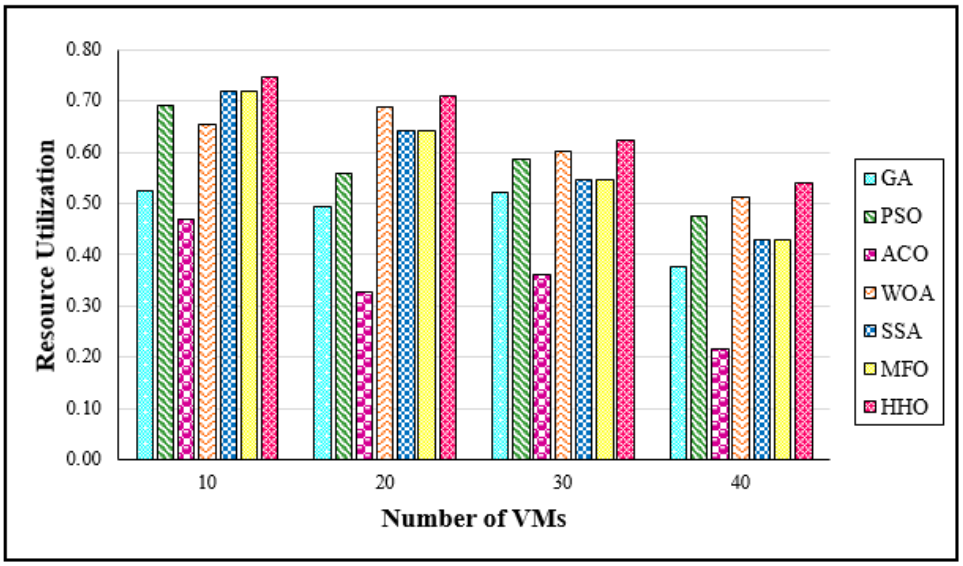

Figure 24: Resource utilization for the different number of VMs.

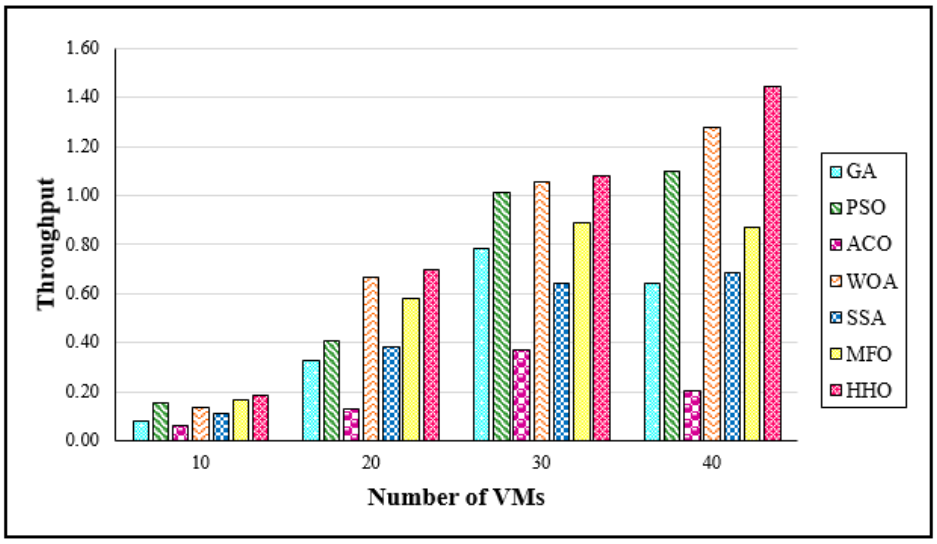

Figure 25: Throughput for the different number of VMs.

resource utilization because the optimal use of resources reduces energy consumption in a server. Figure 19 shows the energy consumption of varying VMs for different algorithms. As shown in Fig. 26, the energy consumption is proportional to the number of VMs in the cloud. As the number of VMs increases, the energy consumption increases. The proposed algorithm decreases energy consumption by up to 29%, 11%, 57%, 6%, 27%, and 18%, compared to GA, PSO, ACO, WOA, SSA, and MFO, respectively. It can be seen that the HHO can perform better for a various number of VMs with a certain number of tasks than other algorithms. This is due to the balance between the exploration and exploitation

capability of the HHO algorithm.

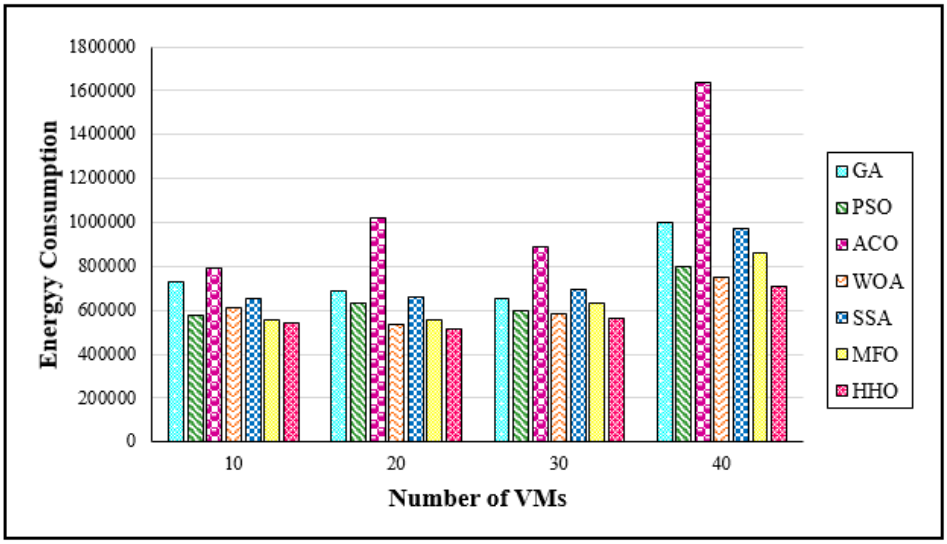

Figure 26: Energy consumption for the different number of VMs.

7.5 Simulation with different number of iterations

In this scenario, the number of input tasks and the number of VMs are fixed (see Table 10).

Table 10: Simulation environment (different number of iterations).

Parameters	**Values**
Number of tasks	500
Tasks size (MI)	100-5000
Task Memory	256-512 MB
Number of VMs	25
VMs execution speed (MIPS)	500-9000
Maximum iteration	100
Population size	60
E_0	(-1, 1)

Figure 27 shows a comparison of seven algorithms with an increasing number of iterations. As can be seen, PSO and WOA work better than GA and ACO, respectively. Because GA and ACO have poor exploitation capability. Also, it can be found that SSA and MFO are more efficient than PSO and WOA because they have better search capability. In addition, it can be found that HHO has the best performance. This is because HHO can find its optimal solution faster than other algorithms. In other words, HHO has much better convergence speed and accuracy than other algorithms. Therefore, it can be found that HHO can have a strong ability to solve complex optimization problems due to its suitable transition from exploration to exploitation.

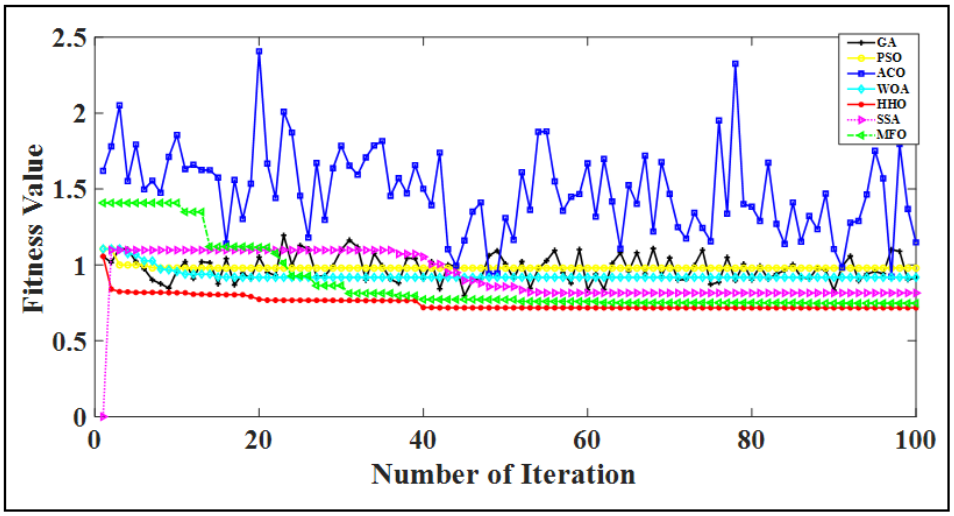

Figure 27: Convergence plot according to the number of iterations.

7.6 Comparison between priority and non-priority scheduling algorithm

In the last scenario, the proposed algorithm is compared with the case that does not take into account the priority. Table 11 shows the setting of parameters.

Table 11: Simulation environment (different number of iterations).

Parameters	**Values**
Number of tasks	800
Tasks size (MI)	100-5000
Task Memory	256-512 MB
Number of VMs	15
VMs execution speed (MIPS)	500-9000
Maximum iteration	100
Population size	40
E_0	(-1, 1)

Figures 28 to 31 show the performance of the task scheduling algorithm with and without priority phase. It can be observed that when tasks are prioritized, the task scheduling algorithm performs better in terms of makespan, resource utilization, throughput, and energy consumption compared to the case where priority is not considered. This is because tasks with less length and memory have higher priority and are performed faster, which improves system efficiency.

8 Conclusion and Future works

Assigning tasks to VMs properly is a challenging issue in the cloud environment. Many algorithms have been presented to optimize the scheduling process in the cloud but existing

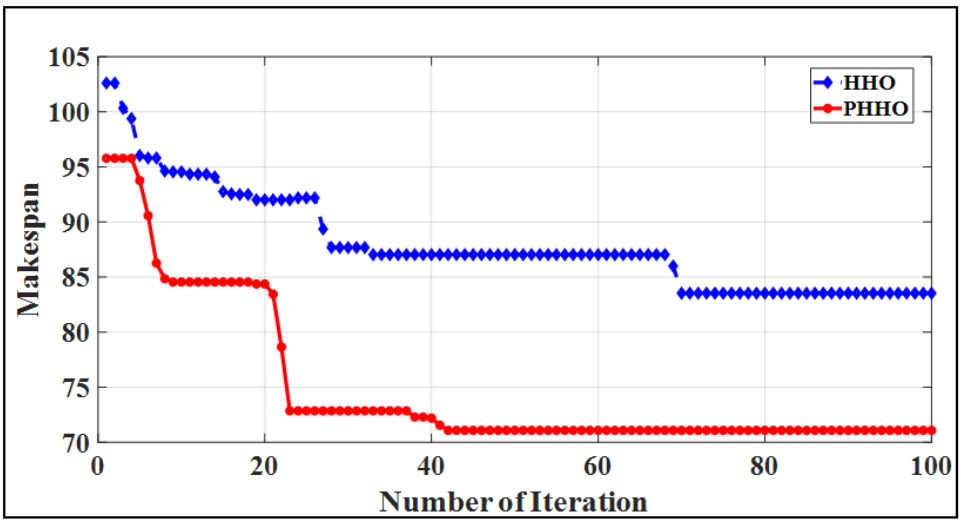

Figure 28: Makespan for PHHO vs. HHO.

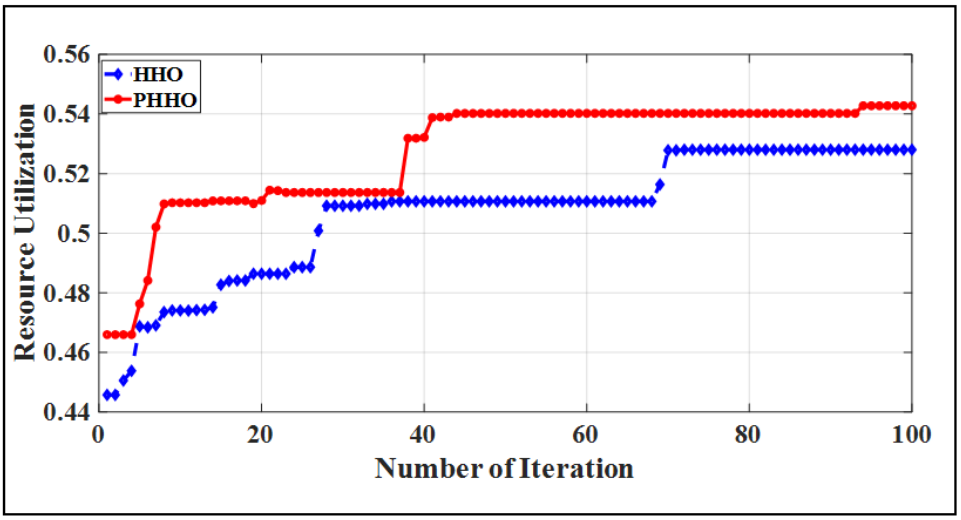

Figure 29: Resource utilization for PHHO vs. HHO.

algorithms usually do not take into account conflicting parameters such as makespan and energy consumption. Therefore, this paper presents an efficient task scheduling algorithm that uses a hierarchical process to prioritize tasks before sending them to the scheduler. Then, it optimally assigns tasks to resources using the new meta-heuristic HHO algorithm. The goal of the proposed algorithm is to make a trade-off between makespan, energy consumption, throughput, and resource utilization. The experimental results showed that the proposed algorithm performed better in terms of makespan, energy consumption, resource utilization, and throughput compared to GA, PSO, WOA, SSA, ACO, and MFO. In addition, the HHO has a better convergence speed compared to other algorithms due to the balance between exploration and exploitation capability. In future studies, the

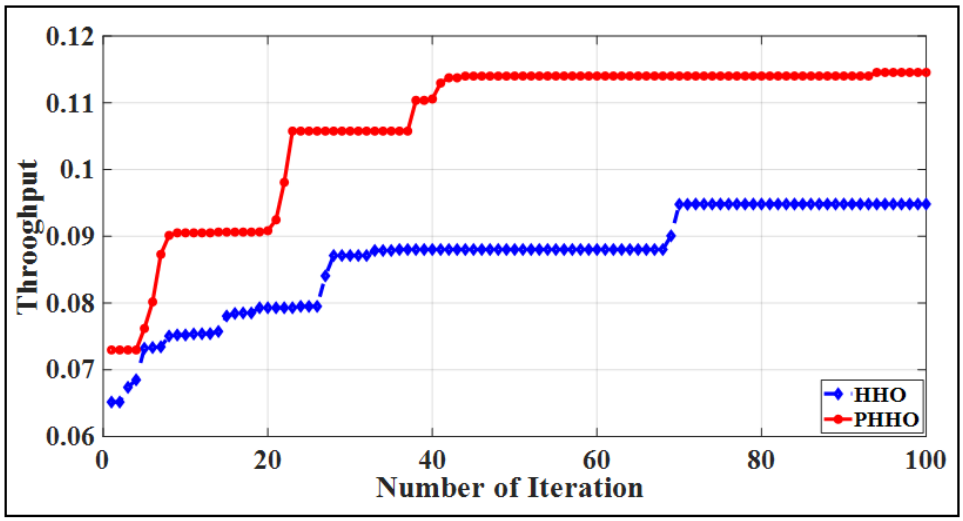

Figure 30: Throughput for PHHO vs. HHO.

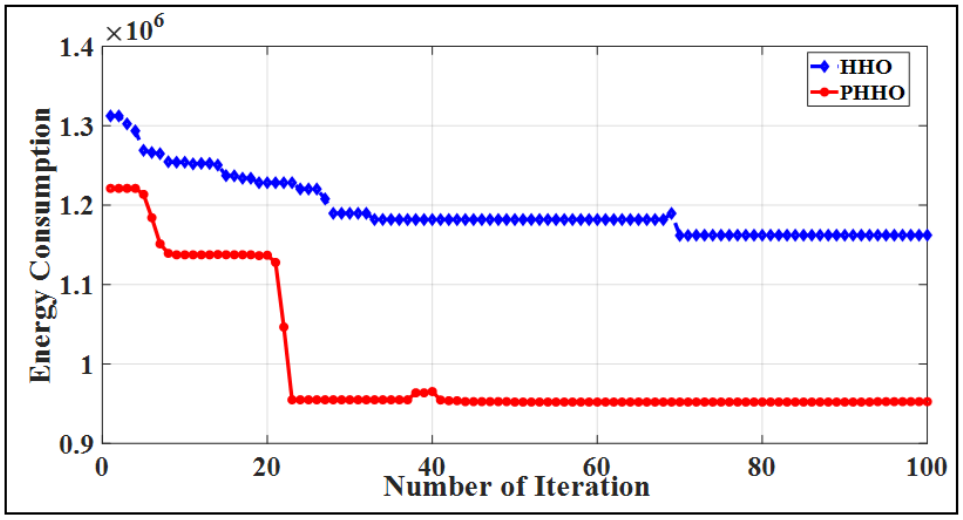

Figure 31: Energy consumption for PHHO vs. HHO.

other QoS parameters like security, and availability need to be applied. We will also use a combination of the HHO algorithm with other meta-heuristic algorithms to improve scheduling efficiency. Finally, to reduce the scheduling overhead of the algorithm for a large workload, the parallel implementation of the algorithm in cloud environments is suggested.

References

[1] Abd Elaziz, M., Xiong, S., Jayasena, K. P. N. and Li, L., Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. Knowledge-Based Systems, 169 (2019) 3952.

[2] Alabool, H., Al- Arabiat, D., Abualigah, L. and Heidari, A. A., Harris hawks optimization: a comprehensive review of recent variants and applications. Neural Computing and Applications, 33 (2021).

[3] Alla, H. Ben, Alla, S. Ben and Ezzati, A., A priority based task scheduling in cloud computing using a hybrid MCDM model. International Symposium on Ubiquitous Networking, Springer, (2017) 235246.

[4] Alsaidy, S. A., Abbood, A. D. and Sahib, M. A., Heuristic initialization of PSO task scheduling algorithm in cloud computing. Journal of King Saud University-Computer and Information Sciences, (2020).

[5] Bello, S. A., Oyedele, L. O., Akinade, O. O., Bilal, M., Delgado, J. M. D., Akanbi, L. A., Ajayi, A. O. and Owolabi, H. A., Cloud computing in construction industry: Use cases, benefits and challenges. Automation in Construction, 122 (2021).

[6] Chen, X., Cheng, L., Liu, C., Liu, Q., Liu, J., Mao, Y. and Murphy, J., A woa-based optimization approach for task scheduling in cloud computing systems. IEEE Systems Journal, 14 (2020) 31173128.

[7] Chen, Z.-Y. and Dai, Z.-H., Application of group decision-making AHP of confidence index and cloud model for rock slope stability evaluation. Computers and Geosciences, (2021).

[8] Dokeroglu, T., Sevinc, E., Kucukyilmaz, T. and Cosar, A., A survey on new generation metaheuristic algorithms. Computers and Industrial Engineering, 137 (2019).

[9] Dorigo, M., Maniezzo, V. and Colorni, A., Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 26 (1996) 2941.

[10] Dos Santos, P. H., Neves, S. M., SantAnna, D. O., de Oliveira, C. H. and Carvalho, H. D., The analytic hierarchy process supporting decision making for sustainable development: An overview of applications. Journal of Cleaner Production, 212 (2019) 119138.

[11] Emami, H., Cloud task scheduling using enhanced sunflower optimization algorithm. ICT Express, (2021).

[12] Er-raji, N., Benabbou, F. and Eddaoui, A., A new task scheduling algorithm for improving tasks execution time in cloud computing. Proceedings of the Mediterranean Symposium on Smart City Applications, Springer, (2017) 298304.

[13] Gao, S., Zhang, Z. and Cao, C., Calculating weights methods in complete matrices and incomplete matrices. Journal of Software, 5 (2010) 304311.

[14] Ghanbari, S. and Othman, M., A priority based job scheduling algorithm in cloud computing. Procedia Engineering, 50 (2012) 778785.

[15] Guo, X., Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm. Alexandria Engineering Journal, 60 (2021) 56035609.

[16] Heidari, A. A., Abbaspour, R. A. and Jordehi, A. R., An efficient chaotic water cycle algorithm for optimization tasks. Neural Computing and Applications, 28 (2017) 5785.

[17] Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M. and Chen, H., Harris hawks optimization: Algorithm and applications. Future Generation Computer Systems, 97 (2019) 849872.

[18] Holland, J. H. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence, MIT press, (1992).

[19] Hosseinzadeh, M., Ghafour, M. Y., Hama, H. K., Vo, B. and Khoshnevis, A., Multi-objective task and workflow scheduling approaches in cloud computing: a comprehensive review. Journal of Grid Computing, 18 (2020) 327356.

[20] Houssein, E. H., Gad, A. G., Wazery, Y. M. and Suganthan, P. N., Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends. Swarm and Evolutionary Computation, 62 (2021).

[21] Jacob, T. P. and Pradeep, K., A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization. Wireless Personal Communications, 109 (2019) 315331.

[22] Jena, U. K., Das, P. K. and Kabat, M. R., Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment. Journal of King Saud University-Computer and Information Sciences (2020).

[23] Kennedy, J. and Eberhart, R., Particle swarm optimization. Proceedings of ICNN95-International Conference on Neural Networks, IEEE, (1995) 19421948.

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

[25] Kumar, A. M. S. and Venkatesan, M., Task scheduling in a cloud computing environment using HGPSO algorithm. Cluster Computing, 22 (2019) 21792185.

[26] Kumar, M., Sharma, S. C., Goel, A. and Singh, S. P., A comprehensive survey for scheduling techniques in cloud computing. Journal of Network and Computer Applications, 143 (2019) 133.

[27] Lin, W., Wang, W., Wu, W., Pang, X., Liu, B. and Zhang, Y., A heuristic task scheduling algorithm based on server power efficiency model in cloud environments. Sustainable Computing: Informatics and Systems, 20 (2018) 5665.

[28] Mansouri, N., Ghafari, R. and Zade, B. M. H., Cloud computing simulators: A comprehensive review. Simulation Modelling Practice and Theory, 104 (2020).

[29] Mansouri, N. and Javidi, M. M., Cost-based job scheduling strategy in cloud computing environments. Distributed and Parallel Databases, 38 (2020) 365400.

[30] Mell, P. and Grance, T., The NIST definition of cloud computing, (2011).

[31] Mirjalili, S., Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowledge-Based Systems, 89 (2015) 228249.

[32] Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H. and Mirjalili, S. M., Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Advances in Engineering Software, 114 (2017) 163191.

[33] Mirjalili, S. and Lewis, A., The whale optimization algorithm. Advances in Engineering Software, 95 (2016) 5167.

[34] Mirjalili, S., Mirjalili, S. M. and Lewis, A., Grey wolf optimizer. Advances in Engineering Software, 69 (2014) 4661.

[35] Mishra, K., Pati, J. and Majhi, S. K., A dynamic load scheduling in IaaS cloud using binary JAYA algorithm. Journal of King Saud University-Computer and Information Sciences, (2020).

[36] Nayak, S. C. and Tripathy, C., Deadline sensitive lease scheduling in cloud computing environment using AHP. Journal of King Saud University-Computer and Information Sciences, 30 (2018) 152163.

[37] Patel, S. J. and Bhoi, U. R., Improved priority based job scheduling algorithm in cloud computing using iterative method. 2014 Fourth International Conference on Advances in Computing and Communications, (2014) 199202.

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

[39] Rajakumar, R., Dhavachelvan, P. and Vengattaraman, T., A survey on nature inspired meta-heuristic algorithms with its domain specifications. 2016 International Conference on Communication and Electronics Systems (ICCES) IEEE, (2016) 16.

[40] Rao, R. V., Savsani, V. J. and Vakharia, D. P., Teachinglearning-based optimization: an optimization method for continuous non-linear large scale problems. Information Sciences, 183 (2012) 115.

[41] Saaty, T. L., How to make a decision: the analytic hierarchy process. European Journal of Operational Research, 48 (1990) 926.

[42] Saaty, T. L., The analytic hierarchy process (AHP). The Journal of the Operational Research Society, 41 (1980) 10731076.

[43] Sanaj, M. S. and Joe Prathap, P. M., Nature inspired chaotic squirrel search algorithm (CSSA) for multi objective task scheduling in an IAAS cloud computing atmosphere. Engineering Science and Technology, an International Journal, 23 (2020) 891902.

[44] Shafiq, D. A., Jhanjhi, N. Z. and Abdullah, A., Load balancing techniques in cloud computing environment: A review. Journal of King Saud University - Computer and Information Sciences, (2021).

[45] Shojafar, M., Javanmardi, S., Abolfazli, S. and Cordeschi, N., FUGE: A joint metaheuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. Cluster Computing, 18 (2015) 829844.

[46] Shukri, S. E., Al-Sayyed, R., Hudaib, A. and Mirjalili, S., Enhanced multi-verse optimizer for task scheduling in cloud computing environments. Expert Systems with Applications, 168 (2021).

[47] Sihwail, R., Omar, K., Ariffin, K. A. Z. and Tubishat, M., Improved harris hawks optimization using elite opposition-based learning and novel search mechanism for feature selection. IEEE Access, 8 (2020).

[48] Simon, D., Biogeography-based optimization. IEEE Transactions on Evolutionary Computation, 12 (2008) 702713.

[49] Singh, H., Tyagi, S. and Kumar, P., Scheduling in cloud computing environment using metaheuristic techniques: A survey. In: Mandal J., Bhattacharya D. (eds) Emerging Technology in Modelling and Graphics. Advances in Intelligent Systems and Computing, Springer, Singapore, (2020) 753-763.

[50] Singh, H., Tyagi, S., Kumar, P., Gill, S. S. and Buyya, R., Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: Analysis, performance evaluation, and future directions. Simulation Modelling Practice and Theory, 111 (2021).

[51] Sreenivasulu, G. and Paramasivam, I., Hybrid optimization algorithm for task scheduling and virtual machine allocation in cloud computing. Evolutionary Intelligence, 14 (2021) 10151022.

[52] Srichandan, S., Ashok Kumar, T. and Bibhudatta, S., Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm. Future Computing and Informatics Journal, 3 (2018) 210230.

[53] Storn, R. and Price, K., Differential evolutiona simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11 (1997) 341359.

[54] Talbi, E.-G. Metaheuristics: From design to implementation. John Wiley and Sons, 74 (2009).

[55] Wang, G.-G., Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Memetic Computing, 10 (2018) 151164.

[56] Wilczyski, A. and Koodziej, J., Modelling and simulation of security-aware task scheduling in cloud computing based on Blockchain technology. Simulation Modelling Practice and Theory, 99 (2020).

[57] Yang, X. and Gandomi, A. H., Bat algorithm: a novel approach for global engineering optimization. Engineering Computations, (2012).

[58] Yang, X.-S., Firefly algorithms for multimodal optimization. International Symposium on Stochastic Algorithms, Springer, (2009) 169178.

[59] Yang, X.-S. and Deb, S., Cuckoo search via Lvy flights. 2009 World Congress on Nature and Biologically Inspired Computing (NaBIC), (2009) 210214.

[60] Yang, X.-S., Karamanoglu, M. and He, X., Flower pollination algorithm: a novel approach for multiobjective optimization. Engineering Optimization, 46 (2014) 12221237.

[61] Zandvakili, A., Mansouri, N. and Javidi, M. M., Signature GOA: A novel comfort zone parameter adjustment using fuzzy signature for task scheduling in cloud environment. Journal of Algorithms and Computation, 53 (2021) 6195.

[62] Zhou, X., Lin, F., Yang, L., Nie, J., Tan, Q., Zeng, W. and Zhang, N., Load balancing prediction method of cloud storage based on analytic hierarchy process and hybrid hierarchical genetic algorithm. SpringerPlus, 5 (2016) 123.