applied sciences

MDPI

Article

SCEHO-IPSO: A Nature-Inspired Meta Heuristic Optimization for Task-Scheduling Policy in Cloud Computing

Kaidala Jayaram Rajashekar [1], Channakrishnaraju [2], Puttamadappa Chaluve Gowda [3,*] and Ananda Babu Jayachandra [4]

[1] Department of Information Science and Engineering, Kalpataru Institute of Technology, Tiptur 572201, India; rajkit2006@kittiptur.ac.in
[2] Department of Computer Science and Engineering, Sri Siddhartha Institute of Technology, Tumakuru 572105, India; rajuck@ssit.edu.in
[3] Department of Electronics and Communication Engineering, Dayananda Sagar University, Bengaluru 560078, India
[4] Department of Information Science and Engineering, Malnad College of Engineering, Hassan 573202, India; abj@mcehassan.ac.in
* Correspondence: puttamadappa-ece@dsu.edu.in

Abstract: Task scheduling is an emerging challenge in cloud platforms and is considered a critical application utilized by the cloud service providers and end users. The main challenge faced by the task scheduler is to identify the optimal resources for the input task. In this research, a Sine Cosine-based Elephant Herding Optimization (SCEHO) algorithm is incorporated with the Improved Particle Swarm Optimization (IPSO) algorithm for enhancing the task scheduling behavior by utilizing parameters like load balancing and resource allocation. The conventional EHO and PSO algorithms are improved utilizing a sine cosine-based clan-updating operator and human group optimizer that improve the algorithm's exploration and exploitation abilities and avoid being trapped in the local optima problem. The efficacy of the SCEHO-IPSO algorithm is analyzed by using performance measures like cost, execution time, makespan, latency, and memory storage. The numerical investigation indicates that the SCEHO-IPSO algorithm has a minimum memory storage of 309 kb, a latency of 1510 ms, and an execution time of 612 ms on the Kafka platform, and the obtained results reveal that the SCEHO-IPSO algorithm outperformed other conventional optimization algorithms. The SCEHO-IPSO algorithm converges faster than the other algorithms in the large search spaces, and it is appropriate for large scheduling issues.

Keywords: cloud computing; elephant herding optimization algorithm; load balancing; particle swarm optimization algorithm; resource allocation; task scheduling

Citation: Rajashekar, K.J.; Channakrishnaraju; Gowda, P.C.; Jayachandra, A.B. SCEHO-IPSO: A Nature-Inspired Meta Heuristic Optimization for Task-Scheduling Policy in Cloud Computing. *Appl. Sci.* **2023**, *13*, 10850. https://doi.org/10.3390/app131910850

Academic Editor: Wenzhong Li

Received: 4 August 2023
Revised: 30 August 2023
Accepted: 27 September 2023
Published: 29 September 2023

1. Introduction

In recent decades, cloud computing has been one of the growing paradigms which dynamically virtualizes resources to provide services over web pages [1,2]. The primary objectives of cloud computing are to achieve high reliability, reputation, throughput, accessibility, scalability, and ease of use [3]. However, effective scheduling of tasks is a main concern which facilitates the execution of tasks utilizing the available resources in cloud computing environments [4]. In cloud computing, the task-scheduling problem has gained more attention among researchers due to the applications and the growth of cloud systems. Generally, the jobs scheduled by a user are assigned to capable devices in cloud systems where every job has consecutive tasks [5]. In cloud platforms, the resources are accessed in two ways: (i) using service-level agreements for resource allocation that act as an interface between the resources and applications, and (ii) brokers or users accessing the resources based on their input [6,7].

The objectives of task scheduling mainly include enhancing load balancing ability and resource utilization and reducing energy consumption and task completion time [8]. Improving the ability of load balancing prevents Virtual Machines (VMs) from resource overload, and the reduction of time for task completion improves the users' experience [9]. In addition to this, task scheduling focuses on many Quality of Service (QoS) factors like scalability, availability, throughput, and response time. The highly appropriate resources are utilized for executing tasks based on user requirements [10,11]. In the present decades, several optimization algorithms have been implemented for task scheduling in cloud computing. The traditional optimization algorithms are ineffective in obtaining optimal task allocation due to a poor convergence rate and search ability. The ineffective task scheduling algorithms increase the execution time of the tasks and reduce the throughput of the cloud systems [12,13]. Therefore, a novel SCEHO-IPSO algorithm is proposed in this manuscript for effective task scheduling. The contributions are outlined below:

- Proposed a SCEHO-IPSO algorithm to resolve task-scheduling problems in the cloud computing platforms. In this scenario, the scheduler effectively ranks user tasks based on execution time and memory details.
- Based on the capacity criteria, the SCEHO-IPSO algorithm determines the efficient VMs to execute tasks in the queue. The SCEHO-IPSO algorithm simultaneously enhances resource utilization and decreases the makespan value.
- The SCEHO-IPSO algorithm optimizes task scheduling by identifying the optimal solutions with better convergence rates. The effectiveness of the SCEHO-IPSO algorithm is analyzed by conducting different experiments. The performance measures cost, execution time, makespan, and latency, and memory storage demonstrates the efficacy of the SCEHO-IPSO algorithm over other optimization algorithms.

This manuscript is arranged in this manner. The research papers on the topic "task scheduling" are surveyed in Section 2. The methodology details, numerical results, and the conclusion of this manuscript are denoted in Sections 3–5, respectively.

2. Literature Survey

Abualigah and Diabat [14] employed an Ant Lion Optimization (ALO) algorithm to maximize resource utilization and minimize makespan in the cloud platforms. In this literature, the traditional ALO algorithm was integrated with elite-based differential evolutions for enhancing an exploration and exploitation ability that avoids the local optima problem. The efficacy of the developed optimization algorithm was analyzed on the real-trace and synthetic databases by utilizing Cloud-Sim. The results demonstrate that the ALO algorithm outperformed the existing optimization algorithms by means of processing time.

Zhou et al. [15] integrated a greedy strategy with the Genetic Algorithm (GA) for optimizing the scheduling of tasks. The developed algorithm's performance was analyzed using dissimilar performance measures such as QoS parameters, average response time, and total completion time. The results showed that the developed algorithm performs more effectively related to the existing task scheduling algorithms. However, the use of elite-based differential evolutions and greedy strategy did not provide optimal solutions to all issues.

In the present scenario, cloud users extensively use cloud-based applications. Kumar and Venkatesan [16] have developed a hybrid optimization algorithm, the Ant Colony Optimization (ACO) algorithm with GA, for the effective handling of cloud users' requests. This study utilized a Utility-Based Scheduler (UBS) for identifying suitable resources and order of the tasks. Here, the ACO algorithm was utilized for enhancing the crossover operation in the GA. The extensive experimental investigation stated that the developed hybrid optimization algorithm obtained superior performance in ensuring QoS parameters and task allocation. The efficacy of the hybrid optimization algorithm was validated in light of throughput, completion time, and response time. However, the integration of the two optimization algorithms increased the time complexity. In addition to this, Mapetu

et al. [17] presented a PSO algorithm with low cost and time complexity for the effective balancing and scheduling of tasks in cloud computing. However, the conventional PSO algorithm had the problem of a poor convergence rate.

Natesan and Chokkalingam [18] presented a mean Grey Wolf Optimization (GWO) algorithm to reduce makespan and energy consumption in cloud computing. The aims of the presented optimization algorithm were analyzed utilizing Cloud-Sim for standard workloads (right- and left-skewed). The GWO algorithm had high time complexity because it needed to perform four operations (attacking, encircling, judging, and searching for prey) for scheduling the tasks.

Shukri et al. [19] integrated two meta-heuristic-based optimization algorithms, the PSO algorithm and Multi-Verse Optimization (MVO) algorithm, for the effective scheduling of tasks in the cloud platforms. The obtained numerical results confirmed the effectiveness of the presented hybrid optimization algorithm, which achieved superior performance in improving resource utilization and reducing makespan time.

Velliangiri et al. [20] combined GA with electro search for enhancing task scheduling in cloud platforms by employing different parameters. Here, the electro search provided the best global optimal solutions and the GA provided the best local optimal solutions. As discussed in the earlier literature, the integration of two optimization algorithms increased the time complexity of the system.

Jacob and Pradeep [21] integrated two optimization algorithms, such as the PSO algorithm and cuckoo search algorithm, that made cloud-computing services faster. The primary objective of this literature study was to decrease the violation rate and makespan. On the other hand, Li and Han [22] implemented a discrete Artificial Bee Colony (ABC) algorithm for flexible task scheduling in the cloud platforms. The experiments conducted on the benchmark instances showed the effectiveness of the presented optimization algorithms, but the hybridization of optimization algorithms increased the time complexity.

Alsaidy et al. [23] employed a PSO algorithm for effective task scheduling. The presented optimization algorithm's performance was evaluated by means of total energy consumption, degree of imbalance, total execution time, and makespan. Sanaj and Prathap [24] developed a chaotic-based squirrel search algorithm for optimal multi-task scheduling in the cloud atmosphere. Correspondingly, Kumar and Venkatesan [25] presented a hybrid task scheduling algorithm in order to solve NP-hard problems in cloud computing. Here, the user tasks were stored in the queue manager, and then the priority was estimated. Based on the estimated priority, the resources were allocated for the task. In this literature, the GA was integrated with the PSO algorithm for scheduling the tasks.

In addition to this, Pang et al. [26], Elaziz et al. [27], Li and Wu [28], Hasan et al. [29], Mansouri et al. [30], and Chandrashekar et al. [31] implemented several optimization algorithms like the GA, moth search algorithm, ACO algorithm, PSO algorithm, modified PSO algorithm, and hybrid weighted ACO algorithm for task scheduling in the cloud platforms. The conventional optimization algorithms have poor convergence speed in multi-objective problem statements and get trapped into local optima problems. To highlight the aforementioned concerns, an effective optimization algorithm named SCEHO-IPSO is proposed for task scheduling in cloud computing environments by considering parameters like load balancing and resource utilization.

3. Methodology

A novel optimization algorithm, SCEHO-IPSO, is introduced for task scheduling. There are several indicators utilized for evaluating the efficacy of the task-scheduling algorithm. The following goals are needed to be achieved for an efficient task-scheduling algorithm:

- Minimization of total cost: According to the user's QoS parameters, the limited total monetary cost states that the SCEHO-IPSO algorithm is efficient.
- Maximization of the QoS parameters: The QoS parameters play a crucial role in cloud computing environments and are utilized to analyze the effectiveness of the SCEHO-IPSO algorithm. The higher QoS is superior, while other parameters remain unchanged.

- Workload balancing: Workload balancing is closely related to the resource utilization rate. If it is an excellent task-scheduling algorithm, the majority of resources should be fully used in cloud environments.
- Minimization of makespan: It represents that the proposed optimization algorithm completes the scheduling of tasks with limited execution time.
- Minimization of latency: Latency is an important measure for evaluating the proposed task-scheduling algorithm. The latency and response time should be limited if it is an excellent task-scheduling algorithm. The flow diagram of the proposed work is mentioned in Figure 1.

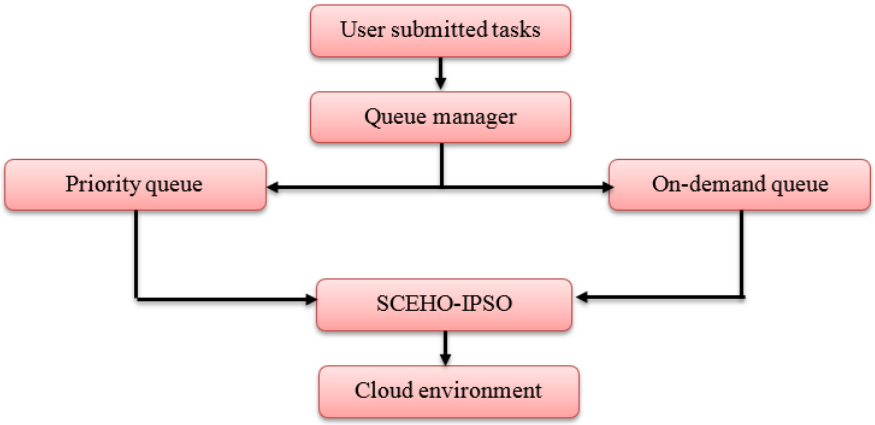

Figure 1. Flow diagram of the proposed work.

3.1. Resource Allocation to the VMs

In this section, resource allocation is considered as the optimization problem, which is mathematically represented in Equation (1).

$$Resource\ allocation = Maximize\left(\frac{P_j^{CPU}}{VM_i^{CPU}} + \frac{P_j^{MEM}}{VM_i^{MEM}} + \frac{P_j^{BW}}{VM_i^{BW}}\right) \quad (1)$$

where P_j^{CPU}, P_j^{MEM}, and P_j^{BW} are denoted as the Computer Processing Unit (CPU), memory, and Bandwidth (BW) of physical machines. Similarly, VM_i^{CPU}, VM_i^{MEM}, and VM_i^{BW} are indicated as the CPU, memory, and bandwidth of the VMs. The proposed optimization algorithm SCEHO-IPSO detects the hosts with higher units based on the following four conditions. The proposed SCEHO-IPSO algorithm detects the hosts with maximum resources.

$$\forall i \sum_{j=1}^{m} y_{ij} = 1 \text{ and } \forall i \sum_{j=1}^{n} y_{ij} VM_i^{CPU} \leq P_j^{CPU}$$

$$\forall i \sum_{j=1}^{n} y_{ij} VM_i^{MEM} \leq P_j^{MEM} \text{ and } \forall i \sum_{j=1}^{n} y_{ij} VM_i^{BW} \leq P_j^{BW}$$

where the binary variable is represented as y_{ij}, the task count is denoted as M, and the number of tasks is indicated as n. The VMs are positioned on appropriate physical machines if the aforementioned conditions are satisfied.

3.2. Load Balancing in the VMs

The VM load status is estimated based on parameters like bandwidth, memory storage, and processor load. These parameters are responsible for pre-determining the VM load status, which is mathematically defined in Equations (2)–(5).

$$L = \{L_1, L_2 \text{ and } L_3\} \quad (2)$$

where

$$L_1 = CPU\ usage\ of\ VM_i / P_j^{CPU} \tag{3}$$

$$L_2 = Memory\ usage\ of\ VM_i / P_j^{MEM} \tag{4}$$

$$L_3 = Bandwidth\ usage\ of\ VM_i / P_j^{BW} \tag{5}$$

where $L_i(t) = \sum_{j=1}^{n} \frac{L_j}{n}$ is the degree of load. If the following two conditions are satisfied—$\left\{ \begin{matrix} idle,\ L_i(t) = L_i^{idle}(t) = 0 \\ under\ load,\ L_i^{idle}(t) < L_i(t) < L_i^{min}(t) \end{matrix} \right\}$ and $\left\{ \begin{matrix} normal,\ L_i^{min}(t) < L_i(t) < L_i^{max}(t) \\ overload, L_i(t) > L_i^{max}(t) \end{matrix} \right\}$—the load status of the VMs is determined. In addition, $L_i^{max}(t)$ and $L_i^{min}(t)$ are represented as the maximum and minimum load in the host.

3.3. Task Scheduling

The process of allocating tasks to the VMs in cloud computing is called task scheduling. In the context of the cloud, the scheduling algorithm maximizes resource usage, reduces the total processing time, saves expenses and energy, and enhances the system's load balance and throughput. In this section, the proposed optimization algorithm, SCEHO-IPSO, satisfies the following three conditions ($\forall i \sum_{j=1}^{m} t_{ij} = 1, F_{t_i} \leq A_{t_i} + D_{t_i}$, and $Et_i + D_{t_i} \leq Ct_i$) for effective task scheduling. The tasks allocated to the jth VM is represented as t_{ij}, F_{t_i} is stated as the finishing time of task t_i, A_{t_i} is represented as the arrival time of task t_i, D_{t_i} is denoted as the dead-line of task t_i, and E_{t_i} and Ct_i are the execution and completion times of task t_i. The mathematical presentation of Ct_i is presented in Equation (6).

$$Ct_i = Et_i + Wt_i \tag{6}$$

where Wt_i is the waiting time of the ith task.

3.4. SCEHO Algorithm

The EHO algorithm is one of the effective metaheuristic-based optimization algorithms which follows the herding behavior of elephants [32,33]. The SCEHO uses a sine cosine-based clan-updating operator for updating the distance between elephants in every clan based on the matriarch elephant's position. The SCEHO algorithm follows three rules in optimization problems: (i) elephants live peacefully in each clan under the leadership of matriarch elephant; (ii) after a specific time period, the male elephants leave their clans and live solely; and (iii) elephants are divided into many clans, where each clan has a fixed population [34–36].

3.4.1. Process of Clan Updating

As discussed earlier, all the elephants live together under the leadership of a matriarch elephant. Generally, the positions of the elephants are influenced by a matriarch elephant based on Equation (7). In this scenario, the sine cosine-based clan-updating operator is employed for updating the clans, which is mathematically determined in Equation (8). The use of the sine cosine-based clan-updating operator enhances the optimization algorithm's exploration and exploitation abilities and avoids being trapped in local optima problems.

$$x_{c_i,j}^{new} = x_{c_i,j}^{t} + \alpha \times \left(x_{best,c_i}^{t} - x_{c_i,j}^{t} \right) \times r1 \tag{7}$$

$$x_{c_i,j}^{new} = \left\{ \begin{matrix} x_{c_i,j}^{t} + r1 \times \sin(r2) \times \left| r3 \times x_{best,c_i}^{t} - x_{c_i,j}^{t} \right| if\ r4 < 0.5 \\ x_{c_i,j}^{t} + r1 \times \cos(r2) \times \left| r3 \times x_{best,c_i}^{t} - x_{c_i,j}^{t} \right| if\ r4 \geq 0.5 \end{matrix} \right\} \tag{8}$$

where α is represented as a scaling factor, which influences the matriarch elephant on $x^t_{c_i,j}$; $x^{new}_{c_i,j}$ are denoted as old and new positions of the jth elephant in clan c_i; x^t_{best,c_i} is stated as a global or best-fitted position of a matriarch elephant in clan c_i; t is indicated as an iteration; and the random numbers $r1, r2, r3,$ and $r4$ perform uniform distribution which ranges between zero and one. On the other hand, the position of a matriarch elephant is updated based on Equation (9).

$$x^{new}_{c_i,j} = \beta \times x^t_{center,c_i} \tag{9}$$

where the term β influences x^t_{center,c_i} on $x^{new}_{c_i,j}$, which ranges between zero and one. The center of the clan x^t_{center,c_i} in the dth dimensional space is mathematically determined in Equation (10).

$$x^t_{center,c_i} = \frac{1}{n_{c_i}} \times \sum_{j=1}^{n_{c_i}} x_{c_i,j,d} \tag{10}$$

where n_{c_i} is represented as the number of elephants in clan c_i. The architecture of the SCEHO algorithm is mentioned in Figure 2.

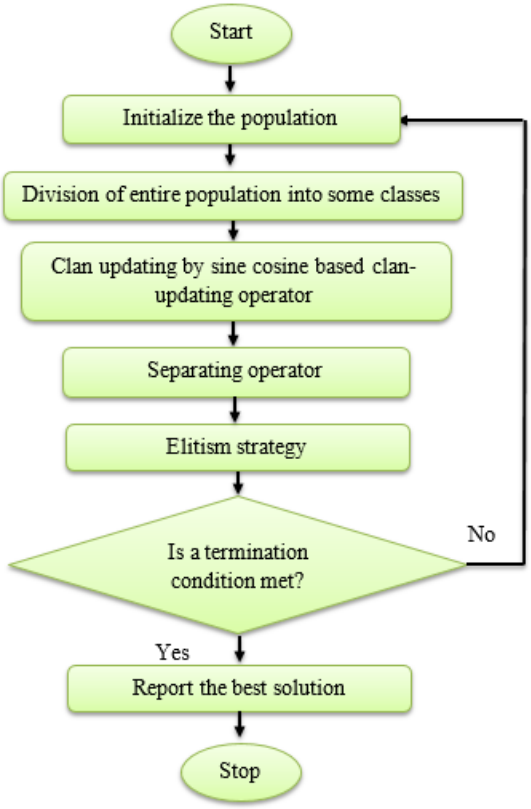

Figure 2. Architecture of SCEHO algorithm.

3.4.2. Process of Separation

When the male elephants attain puberty, they leave the clan and live alone. The process of separation is modeled by a separating operator which helps in resolving the optimization problems. The elephants with the worst fitness are removed by using the separating operator that superiorly enhances the searching ability of the conventional EHO algorithm. The separating operator is a fitness function in this study that is mathematically denoted in Equation (11).

$$x^t_{worst,c_i} = x_{min} + (x_{max} - x_{min} + 1) \times rand \tag{11}$$

where the lower and upper bounds of the elephant position are denoted as x_{min} and x_{max}, x^t_{worst,c_i} is stated as the worst elephants in clan c_i, and the term $rand\epsilon[0,1]$ is represented as the stochastic distribution function. The assumed parameters of the SCEHO algorithm are represented as follows: the iteration number is 100, the population number is 100, $\alpha = 0.5$, the upper bound is 0.9, the number of clans is 10, the set elitism is 2, the lower bound is 0.3, and $\beta = 1$.

3.5. IPSO Algorithm

After finding the best local optimal solutions with the SCEHO algorithm, the best global optimal solutions are determined with the IPSO algorithm. The PSO algorithm is one of the stochastic optimization algorithms which follows swarm movement and intelligence behaviors [37–39]. The social interaction concept is used in the conventional PSO algorithm for resolving the optimization problems. The PSO algorithm makes use of agents (particles) which constitute the swarms that move in the search space [40,41]. For every iteration, the agents (particles) update their positions in order to obtain optimal solutions. In the swarm, every particle moves towards its prior global and personal best position. The Equations (12) and (13) are used to update the velocity and position of the agents (particles). The architecture of the IPSO algorithm is specified in Figure 3.

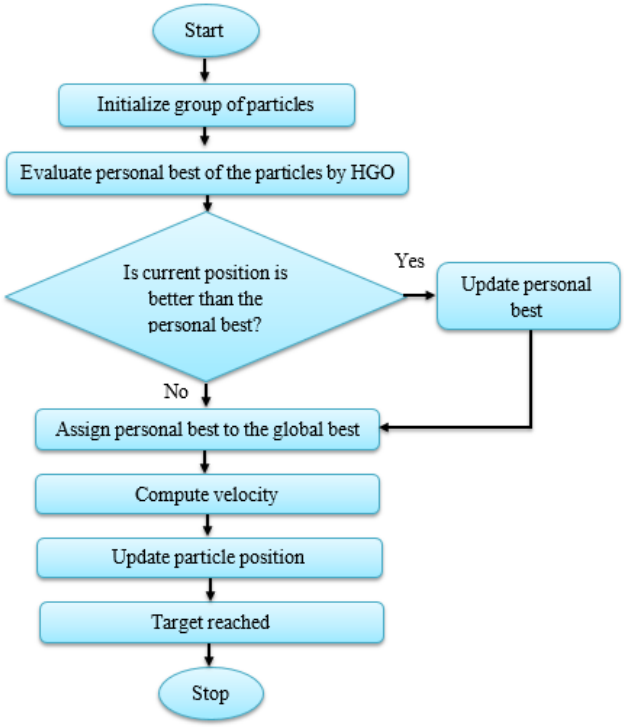

Figure 3. Architecture of IPSO algorithm.

$$v_{id}(t+1) = I_w \times v_{id}(t) + ac_1 \times r_1 \times [p_{id}(t) - o_{id}(t)] + ac_2 \times r_2 \times \left[p_{gd}(t) - o_{id}(t)\right] \quad (12)$$

$$o_{id}(t+1) = o_{id}(t) + v_{id}(t+1) \quad (13)$$

where the random numbers are represented as r_1 and r_2, the acceleration coefficients are denoted as ac_1 and ac_2, and the inertia weight used to balance local and global search is specified as I_w. The particles' global best position and the personal best position are denoted as p_{gd} and p_{id}. In the IPSO algorithm, a novel Human Group Optimizer (HGO) is used to influence the agents (particles). The HGO uses an adaptive uniform mutation operator for enhancing the convergence speed of the conventional PSO algorithm. Additionally, nonlinear function p_m is used in the IPSO algorithm for controlling the range and decision

of the mutation on each particle. The nonlinear function p_m is updated after every iteration and it is mathematically specified in Equation (14).

$$p_m = 0.5 \times e^{(-10 \times \frac{t}{T})} + 0.01 \quad (14)$$

where T indicates the maximum iteration, and t denotes the total number of iterations. The assumed parameters of the IPSO algorithm are listed as follows: the iteration number is 100, the population number is 100, the cognitive constant ac_2 is two, and the social constant ac_1 is three. The numerical analysis of the proposed SCEHO-IPSO algorithm is detailed in Section 4, and the steps involved in the Algorithm 1 are described below:

Algorithm 1. SCEHO-IPSO algorithm.

Step 1: Initialize the objective functions.
Step 2: Create initial population.
Step 3: Evaluate fitness value.
Step 4: For every task, find the best local optimal solutions using SCEHO algorithm.
Step 5: For every task, find the best global optimal solutions using IPSO algorithm.
Step 6: Find the hybrid solutions.
Step 7: If the hybrid new solution value is higher than the current value,
then
Step 8: replace the current value with the hybrid new solution.
Step 9: Select any resources among the population.
Step 10: If the execution time is higher for the selected resource, then eliminate the respective resource and select another resource.
Step 11: Update personal best and global best solutions.
Step 12: Retain it and rank the best solutions.
Step 13: End.

4. Simulation Results

In this manuscript, the SCEHO-IPSO algorithm is simulated utilizing a Cloud-Sim toolkit, which effectively supports on-demand resource provisioning. In addition, it offers a wide range of features, including support for multi-objective optimization scenarios, the dynamic scaling of resources, the modeling of several application characteristics, and an ability to simulate different cloud deployment models. The performance of the SCEHO-IPSO algorithm is analyzed using a system with an Intel core i9 12th generation processor, Linux-operating system, 128 GB random access memory, and 36 GB virtual storage. The load balancing on Kafka is a straightforward and simple process that is managed by the Kafka producers. The efficacy of the SCEHO-IPSO algorithm is compared with other optimization algorithms like ALO, GA, ACO, PSO, GWO, and MVO by means of cost, execution time, makespan, latency, and memory storage. The parameters considered for experimental analysis are mentioned in Table 1.

Table 1. Parameters considered for experimental analysis.

Datacenter	
Number of hosts	2
Number of datacenters	10
VMs	
Number of processing elements	2
Bandwidth	500
Million instructions per seconds	500
Number of VMs	1000
Number of service providers	5
Task (cloud-let)	
Number of tasks	1000
Task length	1000

4.1. Performance Measures

As mentioned earlier, the efficacy of the SCEHO-IPSO algorithm is analyzed by using performance measures like cost, execution time, makespan, latency, and memory storage. At first, the cost represents the total cost (dollars) required for task scheduling in the cloud environments and it is mathematically depicted in Equation (15). Then, makespan is the completion time of the last task to leave the system, and it is mathematically stated in Equation (16). The term F_{t_i} is stated as the finishing time of task t_i, $EC_{t_i r_n}$ is denoted as the execution cost of task t_i on resource r_n, and n is indicated as the number of tasks.

$$Cost = \sum_{i=1}^{n} EC_{t_i r_n} \tag{15}$$

$$Makespan = \sum_{i=1}^{n} F_{t_i} \tag{16}$$

The execution time of the task is defined as the time consumed by the system for executing a specific task. The mathematical formula for execution time E_{t_i} is given in Equation (17). The term P_{P_v} is represented as the processing power of the VMs, S_{t_i} is specified as the task size, and E_{t_i} is indicated as the execution time of task t_i on the VMs.

$$E_{t_i} = \frac{S_{t_i}}{P_{P_v}} \tag{17}$$

The latency is defined as the time consumed for balancing the data load. The effectiveness of the system is improved by decreasing the latency. The mathematical formula to compute latency is specified in Equation (18). The term β_i is represented as the instances generated from the source code, k_j is stated as the number of instances per unit, and $u(t)$ is indicated as the total count of the workload. In addition to this, the memory storage is defined as the space required by the proposed optimization algorithm in order to execute a specific task. The execution time is minimized by reducing the memory storage.

$$Latency = \left(\frac{\beta_i}{k_j}\right)(u(t) + 1) \tag{18}$$

4.2. Quantitative Analysis

The experimental results of different optimization algorithms by means of execution time and cost are presented in Table 2. In addition to this, the performances of the optimization algorithms are analyzed on different platforms such as Storm, Flink, Spark, and Kafka. By inspecting Table 2, the SCEHO-IPSO algorithm is seen to obtain a minimum execution time of 612 milliseconds (ms) and a cost of 62 on the Kafka platform, which are superior compared to other optimization algorithms (ALO, GA, ACO, PSO, GWO, MVO, and EHO) and other platforms (Storm, Flink, and Spark). The two common parameters assumed in the comparative optimization algorithms are a population size of 100 and a maximum iteration of 100. The specific parameters of ALO are as follows: the number of dimensions is five, the lower bound is 0.1, and the upper bound is 0.8. The assumed parameters of GA are as follows: the mutation probability is 0.02, the crossover probability is 0.60, and the number of demes is six. The parameters of ACO are as follows: the time factor is two, the saving matrix factor is two, the visibility coefficient is three, and the pheromone concentration coefficient is one. The maximum initial velocity is 15, the minimal initial velocity is five, the alpha is 0.8, and the beta is 0.8; these are the specific parameters considered in the PSO algorithm. The assumed parameters of the GWO and MVO algorithms are as follows: the number of appliances is 12, the coefficient vector is one, the TDR is one, and the WEP is 0.2. The parameters of EHO are as follows: the alpha is 0.5, the beta is one, the upper bound is 0.9, the number of clans is 10, the set elitism is two, and the lower bound is 0.3. Generally, the highly available fault-tolerant task scheduling helps in improving the business goals. The Kafka platform includes advantages such as high enterprise security, real time analysis,

effective management of clouds, platform scalability, and better processing speed over other platforms. A visual comparison of different optimization algorithms by means of execution time and cost is represented in Figures 4 and 5.

Table 2. Results of different optimization algorithms in light of execution time and cost.

Execution Time (ms)								
Platform	**ALO**	**GA**	**ACO**	**PSO**	**GWO**	**MVO**	**EHO**	**SCEHO-IPSO**
Storm	894	1203	910	887	978	963	834	733
Flink	873	1129	905	876	956	904	820	720
Spark	802	1082	890	864	944	896	802	652
Kafka	772	910	787	793	892	834	772	612
Cost								
Platform	ALO	GA	ACO	PSO	GWO	MVO	EHO	SCEHO-IPSO
Storm	202	190	208	152	188	192	116	102
Flink	193	182	201	144	172	188	102	88
Spark	170	177	188	138	166	177	92	82
Kafka	154	173	177	122	152	152	72	62

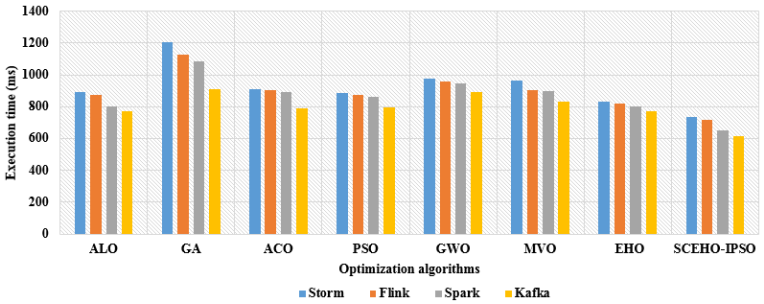

Figure 4. Visual comparison of different optimization algorithms in light of execution time.

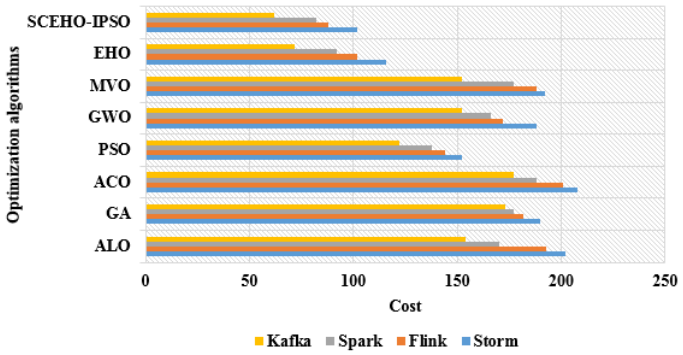

Figure 5. Visual comparison of different optimization algorithms in terms of cost.

The experimental results of different optimization algorithms by means of latency are presented in Table 3. Similar to Table 2, the SCEHO-IPSO algorithm has a lower latency value than conventional optimization algorithms such as ALO, GA, ACO, PSO, GWO, MVO, and EHO. Here, the proposed SCEHO-IPSO algorithm has latency of 1630 ms, 1626 ms, 1550 ms, and 1510 ms on the Storm, Flink, Spark, and Kafka platforms. A visual comparison of different optimization algorithms in terms of latency is depicted in Figure 6.

Table 3. Results of different optimization algorithms in light of latency.

Latency (ms)								
Platform	**ALO**	**GA**	**ACO**	**PSO**	**GWO**	**MVO**	**EHO**	**SCEHO-IPSO**
Storm	2800	2914	3018	3920	2990	2560	1982	1630
Flink	2773	2888	2822	3892	2967	2521	1928	1626
Spark	2822	2880	2902	3620	2940	2490	1802	1550
Kafka	2754	2635	2772	3450	2829	2339	1820	1510

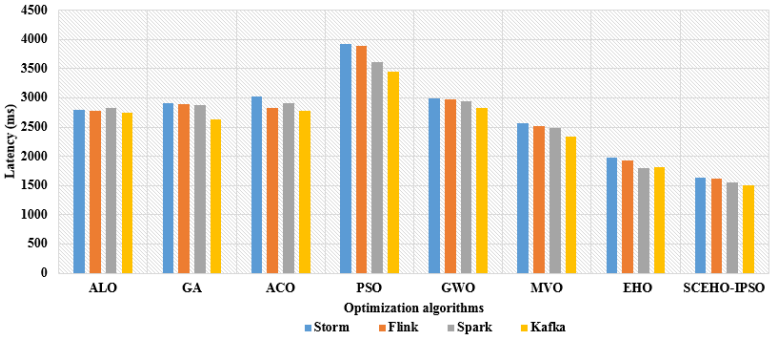

Figure 6. Visual comparison of different optimization algorithms in terms of latency.

Correspondingly, the experimental results of different optimization algorithms in light of makespan and memory storage are specified in Table 4. As seen in Table 4, the proposed SCEHO-IPSO algorithm has minimum makespan values of 88, 73, 45, and 44 on the platforms Storm, Flink, Spark, and Kafka. On the other hand, the proposed SCEHO-IPSO algorithm has a consumed minimal memory storage of 338 kb, 336 kb, 322 kb, and 309 kb on the platforms Storm, Flink, Spark, and Kafka. The achieved results are superior when compared to traditional optimization algorithms such as ALO, GA, ACO, PSO, GWO, MVO, and EHO. Visual comparisons of different optimization algorithms in terms of makespan and memory storage are specified in Figures 7 and 8.

Table 4. Results of different optimization algorithms in light of makespan and memory storage.

Makespan								
Platform	**ALO**	**GA**	**ACO**	**PSO**	**GWO**	**MVO**	**EHO**	**SCEHO-IPSO**
Storm	288	283	193	188	190	144	94	88
Flink	276	279	187	176	158	142	90	73
Spark	244	232	143	165	148	122	88	45
Kafka	240	212	123	142	122	110	78	44
Memory storage (kb)								
Platform	ALO	GA	ACO	PSO	GWO	MVO	EHO	SCEHO-IPSO
Storm	512	538	727	721	573	698	413	338
Flink	532	521	632	658	549	690	479	336
Spark	454	477	553	630	490	532	493	322
Kafka	380	392	442	532	422	504	379	309

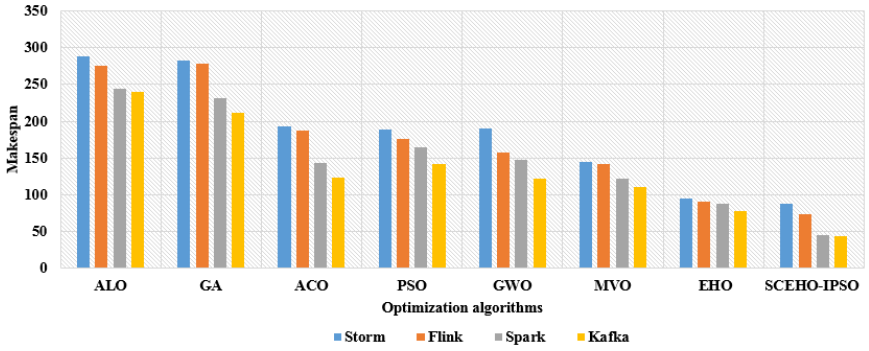

Figure 7. Visual comparison of different optimization algorithms in terms of makespan.

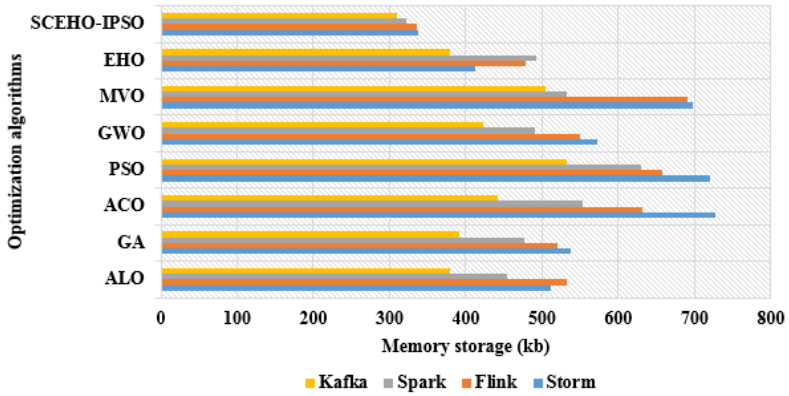

Figure 8. Visual comparison of different optimization algorithms in light of memory storage.

4.3. Discussion

The extensive experimental evaluation shows that the proposed SCEHO-IPSO algorithm has achieved better task scheduling in the cloud platforms by considering parameters like load balancing and resource utilization. The traditional optimization algorithms have poor convergence speed in multi-objective problem statements and become trapped in local optima problems. Compared to existing optimization algorithms like ALO, GA, ACO, PSO, GWO, MVO, and EHO, the proposed SCEHO-IPSO algorithm has the advantages of strong search ability and faster convergence speed, particularly in the context of task scheduling. The performance measures of cost, execution time, makespan, latency, and memory storage demonstrate the efficacy of the proposed SCEHO-IPSO algorithm over other algorithms, which is specifically stated in Tables 2–4. Additionally, related to other platforms, Kafka is extremely fast and massively scalable because it efficiently decouples the data streams, which results in lower latency, cost, execution time, makespan, and sufficient memory storage. The Kafka replicates and distributes partitions across several servers that protect against server failures.

5. Conclusions

In this manuscript, a novel optimization algorithm, SCEHO-IPSO, was implemented for the effective scheduling of tasks in cloud platforms. Here, the proposed SCEHO-IPSO algorithm was implemented using a Cloud-Sim toolkit and was compared with other optimization algorithms like ALO, GA, ACO, PSO, GWO, MVO, and EHO. The proposed SCEHO-IPSO algorithm was analyzed on different platforms, namely Storm, Flink, Spark, and Kafka, and validated by means of cost, execution time, makespan, latency, and memory storage. The extensive experimental investigation states that the proposed SCEHO-IPSO algorithm has a minimum makespan of 44, a memory storage of 309 kb, a latency of 1510 ms, an execution time of 612 ms, and a cost of 62 on the Kafka platform, which are superior to

other optimization algorithms and platforms. Still, the proposed SCEHO-IPSO algorithm has the two major issues of limited energy efficiency and the degree of imbalance.

As a future extension, the proposed optimization algorithm will be implemented in other applications like earth science and climate modeling. Additionally, a hybrid optimization algorithm will be developed and more parameters will be considered for comparisons like energy efficiency and degree of imbalance.

Author Contributions: The paper investigation, resources, data curation, writing—original draft preparation, writing—review and editing, and visualization were performed by K.J.R. The paper conceptualization and software were conducted by C. The validation and formal analysis, methodology, supervision, project administration, and funding acquisition of the version to be published were conducted by P.C.G. and A.B.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

ACO	Ant Colony Optimization
ALO	Ant Lion Optimization
ABC	Artificial Bee Colony
BW	Bandwidth
CPU	Computer Processing Unit
GA	Genetic Algorithm
GWO	Grey Wolf Optimization
HGO	Human Group Optimizer
IPSO	Improved Particle Swarm Optimization
MVO	Multi-Verse Optimization
QoS	Quality of Service
SCEHO	Sine Cosine based Elephant Herding Optimization
UBS	Utility Based Scheduler
VMs	Virtual Machines
Parameters	**Definition**
P_j^{CPU}	Computer processing unit of the physical machines
P_j^{MEM}	Memory of the physical machines
P_j^{BW}	Bandwidth of the physical machines
VM_i^{CPU}	Computer processing unit of the VMs
VM_i^{MEM}	Memory of the VMs
VM_i^{BW}	Bandwidth of the VMs
y_{ij}	Binary variable
M	Task count
n	Number of tasks
$L_i(t)$	Degree of load
$L_i^{max}(t)$	Maximum load in the host
$L_i^{min}(t)$	Minimum load in the host
F_{t_i}	Finishing time of the task t_i,
A_{t_i}	Arrival time of the task t_i,
D_{t_i}	Dead-line of the task t_i
E_{t_i}	Execution time of the task t_i.
Ct_i	Completion time of the task t_i
Wt_i	Waiting time of the ith task

α	Scaling factor
$x^t_{c_i,j}$	Old positions of jth elephant in a clan c_i
$x^{new}_{c_i,j}$	New positions of jth elephant in a clan c_i,
x^t_{best,c_i}	Global or best fitted positions of a matriarch elephant in a clan c_i
$r1, r2, r3,$ and $r4$	Random numbers performs uniform distribution
x_{min} and x_{max}	Lower and upper bounds of the elephant position
x^t_{worst,c_i}	Worst elephants in a clan c_i
ac_1 and ac_2	Acceleration coefficients
I_w	Inertia weight used to balance local and global search
p_{gd} and p_{id}	Particles' global best position and the personal best position
p_m	Nonlinear function
$EC_{t_i r_n}$	Execution cost of the task t_i on a resource r_n
P_{P_v}	Processing power of the VMs
S_{t_i}	Task size
β_i	Instances generated from the source code
k_j	Number of instances per unit
$u(t)$	Total count of the workload

References

1. Jana, B.; Chakraborty, M.; Mandal, T. A task scheduling technique based on particle swarm optimization algorithm in cloud environment. In *Soft Computing: Theories and Applications, Advances in Intelligent Systems and Computing*; Ray, K., Sharma, T., Rawat, S., Saini, R., Bandyopadhyay, A., Eds.; Springer: Singapore, 2019; Volume 742, pp. 525–536. [CrossRef]
2. Houssein, E.H.; Gad, A.G.; Wazery, Y.M.; Suganthan, P.N. Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends. *Swarm Evol. Comput.* **2021**, *62*, 100841. [CrossRef]
3. Arunarani, A.R.; Manjula, D.; Sugumaran, V. Task scheduling techniques in cloud computing: A literature survey. *Future Gener. Comput. Syst.* **2019**, *91*, 407–415. [CrossRef]
4. Gupta, S.; Iyer, S.; Agarwal, G.; Manoharan, P.; Algarni, A.D.; Aldehim, G.; Raahemifar, K. Efficient Prioritization and Processor Selection Schemes for HEFT Algorithm: A Makespan Optimizer for Task Scheduling in Cloud Environment. *Electronics* **2022**, *11*, 2557. [CrossRef]
5. Guo, X. Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm. *Alex. Eng. J.* **2021**, *60*, 5603–5609. [CrossRef]
6. Ding, D.; Fan, X.; Zhao, Y.; Kang, K.; Yin, Q.; Zeng, J. Q-learning based dynamic task scheduling for energy-efficient cloud computing. *Future Gener. Comput. Syst.* **2020**, *108*, 361–371. [CrossRef]
7. Hussain, M.; Wei, L.-F.; Lakhan, A.; Wali, S.; Ali, S.; Hussain, A. Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing. *Sustain. Comput. Inform. Syst.* **2021**, *30*, 100517. [CrossRef]
8. Cai, X.; Geng, S.; Wu, D.; Cai, J.; Chen, J. A multicloud-model-based many-objective intelligent algorithm for efficient task scheduling in internet of things. *IEEE Internet Things J.* **2021**, *8*, 9645–9653. [CrossRef]
9. Bezdan, T.; Zivkovic, M.; Tuba, E.; Strumberger, I.; Bacanin, N.; Tuba, M. Multi-objective Task Scheduling in Cloud Computing Environment by Hybridized Bat Algorithm. In *Intelligent and Fuzzy Techniques: Smart and Innovative Solutions. INFUS 2020. Advances in Intelligent Systems and Computing*; Kahraman, C., Cevik Onar, S., Oztaysi, B., Sari, I., Cebi, S., Tolga, A., Eds.; Springer: Cham, Switzerland, 2021; Volume 1197, pp. 718–725. [CrossRef]
10. Yuan, H.; Bi, J.; Zhou, M.; Liu, Q.; Ammari, A.C. Biobjective task scheduling for distributed green data centers. *IEEE Trans. Autom. Sci. Eng.* **2021**, *18*, 731–742. [CrossRef]
11. Zhou, J.; Sun, J.; Cong, P.; Liu, Z.; Zhou, X.; Wei, T.; Hu, S. Security-critical energy-aware task scheduling for heterogeneous real-time MPSoCs in IoT. *IEEE Trans. Serv. Comput.* **2020**, *13*, 745–758. [CrossRef]
12. Wang, J.; Li, D. Task Scheduling Based on a Hybrid Heuristic Algorithm for Smart Production Line with Fog Computing. *Sensors* **2019**, *19*, 1023. [CrossRef]
13. Boveiri, H.R.; Khayami, R.; Elhoseny, M.; Gunasekaran, M. An efficient Swarm-Intelligence approach for task scheduling in cloud-based internet of things applications. *J. Ambient Intell. Hum. Comput.* **2019**, *10*, 3469–3479. [CrossRef]
14. Abualigah, L.; Diabat, A. A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. *Clust. Comput.* **2021**, *24*, 205–223. [CrossRef]
15. Zhou, Z.; Li, F.; Zhu, H.; Xie, H.; Abawajy, J.H.; Chowdhury, M.U. An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. *Neural Comput. Appl.* **2020**, *32*, 1531–1541. [CrossRef]
16. Kumar, A.M.S.; Venkatesan, M. Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment. *Wirel. Pers. Commun.* **2019**, *107*, 1835–1848. [CrossRef]
17. Mapetu, J.P.B.; Chen, Z.; Kong, L. Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing. *Appl. Intell.* **2019**, *49*, 3308–3330. [CrossRef]

18. Natesan, G.; Chokkalingam, A. Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm. *ICT Express* **2019**, *5*, 110–114. [CrossRef]
19. Shukri, S.E.; Al-Sayyed, R.; Hudaib, A.; Mirjalili, S. Enhanced multi-verse optimizer for task scheduling in cloud computing environments. *Expert Syst. Appl.* **2021**, *168*, 114230. [CrossRef]
20. Velliangiri, S.; Karthikeyan, P.; Xavier, V.M.A.; Baswaraj, D. Hybrid electro search with genetic algorithm for task scheduling in cloud computing. *Ain Shams Eng. J.* **2021**, *12*, 631–639. [CrossRef]
21. Jacob, T.P.; Pradeep, K. A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization. *Wirel. Pers. Commun.* **2019**, *109*, 315–331. [CrossRef]
22. Li, J.; Han, Y. A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system. *Clust. Comput.* **2020**, *23*, 2483–2499. [CrossRef]
23. Alsaidy, S.A.; Abbood, A.D.; Sahib, M.A. Heuristic initialization of PSO task scheduling algorithm in cloud computing. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 2370–2382. [CrossRef]
24. Sanaj, M.S.; Prathap, P.M.J. Nature inspired chaotic squirrel search algorithm (CSSA) for multi objective task scheduling in an IAAS cloud computing atmosphere. *Eng. Sci. Technol. Int. J.* **2020**, *23*, 891–902. [CrossRef]
25. Kumar, A.M.S.; Venkatesan, M. Task scheduling in a cloud computing environment using HGPSO algorithm. *Clust. Comput.* **2019**, *22* (Suppl. 1), 2179–2185. [CrossRef]
26. Pang, S.; Li, W.; He, H.; Shan, Z.; Wang, X. An EDA-GA hybrid algorithm for multi-objective task scheduling in cloud computing. *IEEE Access* **2019**, *7*, 146379–146389. [CrossRef]
27. Elaziz, M.A.; Xiong, S.; Jayasena, K.P.N.; Li, L. Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. *Knowl.-Based Syst.* **2019**, *169*, 39–52. [CrossRef]
28. Li, G.; Wu, Z. Ant Colony Optimization Task Scheduling Algorithm for SWIM Based on Load Balancing. *Future Internet* **2019**, *11*, 90. [CrossRef]
29. Hasan, M.Z.; Al-Rizzo, H.; Al-Turjman, F.; Rodriguez, J.; Radwan, A. Internet of Things Task Scheduling in Cloud Environment Using Particle Swarm Optimization. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6. [CrossRef]
30. Mansouri, N.; Zade, B.M.H.; Javidi, M.M. Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory. *Comput. Ind. Eng.* **2019**, *130*, 597–633. [CrossRef]
31. Chandrashekar, C.; Krishnadoss, P.; Kedalu Poornachary, V.; Ananthakrishnan, B.; Rangasamy, K. HWACOA Scheduler: Hybrid Weighted Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing. *Appl. Sci.* **2023**, *13*, 3433. [CrossRef]
32. Li, J.; Lei, H.; Alavi, A.H.; Wang, G.-G. Elephant Herding Optimization: Variants, Hybrids, and Applications. *Mathematics* **2020**, *8*, 1415. [CrossRef]
33. Li, W.; Wang, G.-G.; Alavi, A.H. Learning-based elephant herding optimization algorithm for solving numerical optimization problems. *Knowl.-Based Syst.* **2020**, *195*, 105675. [CrossRef]
34. Li, W.; Wang, G.-G. Elephant herding optimization using dynamic topology and biogeography-based optimization based on learning for numerical optimization. *Eng. Comput.* **2022**, *38* (Suppl. 2), 1585–1613. [CrossRef]
35. Strumberger, I.; Minovic, M.; Tuba, M.; Bacanin, N. Performance of Elephant Herding Optimization and Tree Growth Algorithm Adapted for Node Localization in Wireless Sensor Networks. *Sensors* **2019**, *19*, 2515. [CrossRef] [PubMed]
36. Elhosseini, M.A.; El Sehiemy, R.A.; Rashwan, Y.I.; Gao, X.Z. On the performance improvement of elephant herding optimization algorithm. *Knowl.-Based Syst.* **2019**, *166*, 58–70. [CrossRef]
37. Wang, X.; Yao, W. A Discrete Particle Swarm Optimization Algorithm for Dynamic Scheduling of Transmission Tasks. *Appl. Sci.* **2023**, *13*, 4353. [CrossRef]
38. Pradhan, A.; Bisoy, S.K.; Das, A. A survey on PSO based meta-heuristic scheduling mechanism in cloud computing environment. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 4888–4901. [CrossRef]
39. Ramírez-Ochoa, D.-D.; Pérez-Domínguez, L.A.; Martínez-Gómez, E.-A.; Luviano-Cruz, D. PSO, a Swarm Intelligence-Based Evolutionary Algorithm as a Decision-Making Strategy: A Review. *Symmetry* **2022**, *14*, 455. [CrossRef]
40. Jiang, H.; He, Z.; Ye, G.; Zhang, H. Network intrusion detection based on PSO-XGBoost model. *IEEE Access* **2020**, *8*, 58392–58401. [CrossRef]
41. Deng, W.; Xu, J.; Zhao, H.; Song, Y. A novel gate resource allocation method using improved PSO-based QEA. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 1737–1745. [CrossRef]