Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.073171

ARTICLE

Energy Aware Task Scheduling of IoT Application Using a Hybrid Metaheuristic Algorithm in Cloud Computing

Ahmed Awad Mohamed[1], Eslam Abdelhakim Seyam[2,*], Ahmed R. Elsaeed[3], Laith Abualigah[4], Aseel Smerat[5,6], Ahmed M. AbdelMouty[7] and Hosam E. Refaat[8]

[1]Information System Department, Cairo Higher Institute for Languages and Simultaneous Interpretation, and Administrative Science, Cairo, 11765, Egypt

[2]Department of Insurance and Risk Management, College of Business, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, 13318, Saudi Arabia

[3]Department of Mathematics and Statistics, Faculty of Science, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, 11432, Saudi Arabia

[4]Computer Science Department, Al-AlBayt University, Mafraq, 25113, Jordan

[5]Faculty of Educational Sciences, Al-Ahliyya Amman University, Amman, 19328, Jordan

[6]Centre for Research Impact and Outcome, Chitkara University, Punjab, 140401, India

[7]Information System Department, Faculty of Computers and Information, Zagazig University, Zagazig, 44519, Egypt

[8]Information System Department, Faculty of Computers and Information, Suez Canal University, Ismailia, 41522, Egypt

*Corresponding Author: Eslam Abdelhakim Seyam. Email: isiam@imamu.edu.sa

Received: 12 September 2025; Accepted: 31 October 2025; Published: 12 January 2026

ABSTRACT: In recent years, fog computing has become an important environment for dealing with the Internet of Things. Fog computing was developed to handle large-scale big data by scheduling tasks via cloud computing. Task scheduling is crucial for efficiently handling IoT user requests, thereby improving system performance, cost, and energy consumption across nodes in cloud computing. With the large amount of data and user requests, achieving the optimal solution to the task scheduling problem is challenging, particularly in terms of cost and energy efficiency. In this paper, we develop novel strategies to save energy consumption across nodes in fog computing when users execute tasks through the least-cost paths. Task scheduling is developed using modified artificial ecosystem optimization (AEO), combined with negative swarm operators, Salp Swarm Algorithm (SSA), in order to competitively optimize their capabilities during the exploitation phase of the optimal search process. In addition, the proposed strategy, Enhancement Artificial Ecosystem Optimization Salp Swarm Algorithm (EAEOSSA), attempts to find the most suitable solution. The optimization that combines cost and energy for multi-objective task scheduling optimization problems. The backpack problem is also added to improve both cost and energy in the iFogSim implementation as well. A comparison was made between the proposed strategy and other strategies in terms of time, cost, energy, and productivity. Experimental results showed that the proposed strategy improved energy consumption, cost, and time over other algorithms. Simulation results demonstrate that the proposed algorithm increases the average cost, average energy consumption, and mean service time in most scenarios, with average reductions of up to 21.15% in cost and 25.8% in energy consumption.

KEYWORDS: Energy-efficient tasks; internet of things (IoT); cloud fog computing; artificial ecosystem-based optimization; salp swarm algorithm; cloud computing

1 Introduction

In recent years, there has been a trend in cloud computing due to its excellence in data distribution, storage efficiency, pay-per-use, and user cost savings. Users are also interested in saving time and effort, as well as reducing costs, during various scheduling tasks on virtual machines. The Internet of Things (IoT) has also been developed and integrated with cloud computing to address the increasing volume of data, which poses a new challenge in cloud computing environments. Cloud computing also provides numerous resources to support a wide range of services across the Internet [1–5]. There are also different policies between users and the level of service in cloud computing, allowing for the increase or decrease of devices according to usage. The Internet of Things (IoT) comprises a network of various devices and sensors that exchange data through cloud computing. It is used in smart cities, smart homes, hospitals, factories, the army, agriculture, and other data-intensive applications. The Internet of Things suffers from excessive energy and storage consumption, so cloud computing resources are necessary to mitigate storage issues. In addition, when cloud computing is combined with the Internet of Things, huge amounts of various data can be stored daily in cloud computing. Fog computing comprises three distinct layers: the cloud computing layer, the first upper layer, the second middle layer, which is fog computing, and the last, lower layer, comprising various devices from the Asian Internet, sensors, radio frequencies, and users. The task scheduling problem in fog computing is an NP-hard problem that requires extensive study and the application of modern metaheuristic algorithms. When tasks are distributed from users to different virtual machines, they have specific and different capacities and capabilities. Therefore, there must be an effective way to schedule different tasks while achieving many goals [6–10]. There is also another important aspect of fog computing, which is the energy consumption in cloud computing when sending tasks and large amounts of data distributed across nodes. It is also one of the important issues that must be solved when transferring data across nodes in cloud computing.

Previous studies have applied a range of different objectives and various artificial intelligence algorithms, revealing weaknesses in previous solutions, such as rapid convergence and the risk of falling into a local optimal point [11–15]. To address these issues, we hybridized two algorithms in this research to mitigate the defects of the previous algorithms. Reducing energy consumption and cost in data centers is a significant challenge and a complex issue, particularly for multitasking operations in cloud computing. To address this problem, several techniques have been implemented to reduce energy usage, such as load balancing. The primary goal is to distribute tasks across virtual machines in a flexible manner, thereby saving both time and cost [16–20]. Energy-efficient task scheduling can be implemented as a dynamic load-balancing mechanism, which distributes workloads based on the speed capacity of virtual machines in cloud computing. Costs have also been modeled using the knapsack problem to minimize costs and select virtual machines directly, thereby saving time and reducing costs [21,22]. This paper presents a new optimization approach, called EAEOSSA, which combines the AEO algorithm with SSA to enhance multi-objective task operations within cloud computing. When the strategy is executed, it competes effectively in the exploration phase; however, in the exploitation phase, the SSA algorithm is employed due to its superior performance. Balance is achieved when sending different tasks and reaching the maximum level, while maintaining the convergence speed between exploitation and extraction capabilities. The primary motivation of this research is to address the challenges of scheduling tasks on fog computing with IoT, including time, cost, energy consumption, node balance, and throughput improvement. Two different algorithms were also combined to achieve the goals of minimizing time, cost, energy consumption, and balancing node loads in cloud computing.

This article made the following contributions:

– A hybrid strategy has been developed between AEO and SSA that explores and exploits the search space locally and globally to optimize various objectives in IoT in cloud computing.

- A low-time solution for sending tasks between nodes to reduce users' waiting time, improve productivity and cost, and reduce energy consumption.
- This work is the first attempt to study a new hybrid AEO, SSA (EAEOSSA), solved to obtain low energy consumption and cost. Using the knapsack problem, to the best of our knowledge, this is the first attempt to implement this strategy.
- The efficiency evaluation of the EAEOSSA strategy is performed through a different set of tasks and several different virtual machines and methods.

The rest of this paper is organized as follows. Section 2 presents the review work. Section 3 shows the system architecture. Section 4 presents the proposed system using AEO and SSA. Section 5 presents the experimental results, and Section 6 concludes with a summary and future work.

2 Related Work

Ref. [23] presented a new scheme to optimize energy, cost, and time and called the proposed algorithm fuzzy-based task scheduling (Security-Aware and Energy-Aware, SAEA) algorithm. It also considered the security levels of the schedule sent by the users and the degrees of balance between the nodes. He also added a set of different goals for implementing the proposed algorithm via cloud computing, which are (i.e., execution time, energy consumption, makespan, degree of imbalance, and security). The results showed the superiority of the proposed algorithm over other algorithms [23].

An energy-efficient resource scheduling algorithm in the Geographically Cloud Computing (GCC) environment is proposed using the Cultural Emperor Penguin Optimizer (CEPO) algorithm, referred to as EERS-CEPO. To improve energy consumption load distribution and reduce delay in responding to users. Two different algorithms were also integrated to enhance the exploration and exploitation capabilities of the proposed algorithm. The results demonstrated the superiority of the proposed algorithm over other algorithms in terms of time, cost, and resource utilization [24].

A new algorithm, called the Dynamic Voltage and Frequency Scaling (DVFS) method [25], is introduced, which prioritizes tasks sent by users to execute them quickly and accurately. The algorithm also classifies virtual devices near users and assigns them the same priority. The proposed algorithm also reduces the cost. The proposed algorithm ensures the execution and migration of jobs, achieving balance within the proposed system. The results also demonstrated a 12% energy improvement over other algorithms.

This work presents different algorithms to test the issue of energy consumption and various resource allocation in cloud computing. It also considered load balancing between nodes to increase the number of different tasks from users. He also compared several different algorithms, and the results showed that the whale algorithm outperforms other algorithms in terms of energy and node balance [26].

This paper presents an algorithm called the Hybrid Genetic Algorithm (HGA) to reduce energy consumption and facilitate the real-time execution of tasks without precedence constraints. It also improved the balance between nodes for faster task execution and response. The simulation results demonstrated the efficiency of the proposed algorithm compared to other algorithms [27].

This paper presents an algorithm called a Cost and Energy-aware Task Scheduling Algorithm (CETSA). This proposed algorithm reduces time, cost, and energy consumption, and considers the overhead between devices to distribute loads across virtual machines more efficiently. Experimental results showed reduced time, cost, energy consumption, and improved load balancing across nodes, as well as its superiority over other algorithms [28].

This paper presented an algorithm called the Discrete Pathfinder Algorithm (DPFA). He presented a set of objectives, including time, energy, and throughput, to minimize cost and energy consumption

across nodes in cloud computing. It also considered the exploitation and exploration stages of the proposed algorithm. The proposed results demonstrated the superiority of the algorithm over other algorithms in terms of time, energy, and productivity [29].

Discussion between Algorithms

Table 1 presents a comparison between the proposed strategy and previous algorithms in terms of their advantages and limitations in cloud computing.

Table 1: Comparison between related work and my strategy

Strategy	Year	Main objective	Hybrid	System	Fog computing based on IoT	Knapsack problem
[23]	2021	Optimize energy, cost, and time	Yes	Hemogenous	Yes	NO
[24]	2023	Reduce delay and improve resource utilization	Yes	Hemogenous	Yes	NO
[25]	2022	Prioritize and execute tasks efficiently	Yes	Hemogenous	Yes	NO
[26]	2021	Balance load and reduce energy consumption	Yes	Hemogenous	Yes	NO
[27]	2017	Reduce energy and achieve real-time scheduling	Yes	Hemogenous	Yes	NO
[28]	2022	Minimize time, cost, and energy	Yes	Hemogenous	Yes	NO
[29]	2021	Optimize throughput, energy, and cost	Yes	Hemogenous	Yes	NO
My Strategy	2025	Optimize throughput, resources, energy, knapsack problem, and cost	Yes	heterogeneous	Yes	Yes

3 Suggested System Model

The problem of scheduling tasks when sending tasks from users on virtual machines across cloud computing nodes is deciding and assigning tasks. Additionally, the system is homogeneous, while the resources of the data centers are heterogeneous, which contributes to the difficulty of establishing a scheduling policy across nodes in cloud computing. The source of concern that affects users and their relationship with the cloud provider is the performance of the cloud system and the availability of resources in terms of cost, energy, and other factors. Users send tasks to different geographical locations through the queue, and a data center broker enters to manage all the tasks sent by users and provide all information about each virtual machine located within the different data centers. The primary goal is to achieve one or more objectives that meet user requirements, such as power, cost, balance, space, and bandwidth reduction. Additionally, one of the goals addressed by this proposed strategy is to implement tasks as efficiently as possible and minimize user waiting time. Typically, when tasks are executed to reach different locations in hierarchical clouds, it requires a long time, consumes large bandwidth, incurs high access costs, and results in energy

waste. Therefore, when sending scheduling tasks across different sites, decisions must be made to reach the necessary sites to accomplish the tasks required for users, while also reducing costs and energy. In the proposed system, EAEOSSA, when executing tasks sent by users, gathers sufficient information about different geographical locations to reduce the need for searching virtual devices across locations in cloud computing. The task manager directs the required tasks to virtual machines via cloud computing based on the proposed EAEOSSA strategy to allocate different tasks and reduce time and cost.

Our proposed method divides the cloud environment into distinct geographical locations and comprises multiple data centers through cloud computing. Different data centers contain a group of physical machines and a group of virtual machines inside them. Although the system is different across cloud computing, it is not homogeneous. When sending various tasks to the nodes via cloud computing, it selects the required locations to minimize time, cost, and the speed of task completion, while also reducing bandwidth according to the proposed EAEOSSA strategy. The elements of the proposed system can be represented as follows:

In Fig. 1, each node can be offered in DC form {dc1, dc2,. . ., dcn}, where n is a set of various nodes in the fog. The physical machine can be represented as PM D fpm1; pm2;. . .; pmx g, where x is a set of various PMs existing in nodes. A virtual machine can be presented as VM D, where fvm1, vm2,. . ., vms is a set of VMs placed on a physical machine. Let T = {T1, T2, T3,..., Tv} be the set of IoT scheduling tasks to be sent to nodes.

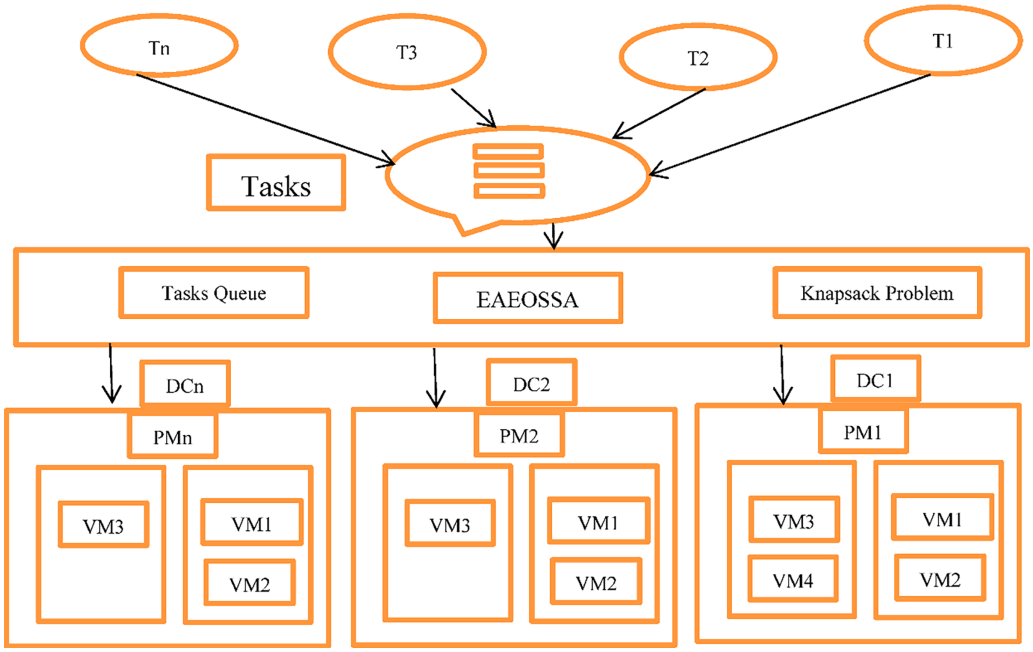


Figure 1: Architecture diagram of the proposed EAEOSSA system with Knapsack Optimization

A new strategy of state-of-the-art algorithms, AEO and SSA, is proposed to reduce network load, energy consumption, and task execution time across different geographical locations. The proposed strategy reduces time, cost, service time, power consumption, and bandwidth. This aims to reduce workloads and automate task assignments from users via the Internet of Things. The proposed system, utilizing EAEOSSA techniques, can reduce time and energy consumption, lower costs, and achieve high optimal utility by implementing IFogSim [25–29].

Artificial ecosystem-based optimization (AEO)

This section presents the AEO algorithm and its associated equations, which are used in the strategy proposed in this research. The algorithm simulates the transfer of energy between objects in general. It consists of three stages to improve research problems: production, consumption, and decomposition. In the first stage of production, it works to achieve a balance between exploration and exploitation during maximization. The second operator, which is consumption, is based on exploring the search space, and the third operator, which is decomposition, is responsible for exploitation and enhancement.

In [30], the algorithm selects the production phase as the worst solution among the solutions found in the different search phases. The decomposition agent intervenes to optimize the positions of the product factor according to different positions and chooses the best individual and the best position. The equation can be formulated as follows:

$$X_1(t+1) = (1-d)\,X_b(t) + d.X_{rand}(t) \tag{1}$$

where

$$d = \left(1 - \frac{t}{t_{max}}\right) rand_1 \tag{2}$$

$$X_{rand}(t) = rand_2(\,) \cdot (ub - lb) + lb \tag{3}$$

In the proposed task scheduling model for cloud computing, cost, energy efficiency and optimal resource utilization are achieved through adaptive position updating by metaheuristic techniques. *ub* and *lb* are the limits of the search domain. $rand_1$ and $rand_2$ denote random values, and *d* refers to a weight coefficient. The maximum iteration number is T. d random number ranging between [0, 1].

The second stage, consumption, indicates that the producer and consumer have low energy as food for other consumers. Each consumer has their function and mission through herbivores, omnivores, and carnivores. the low-energy tasks (consumers) adjust their resource allocation in response to the most energy-efficient virtual machines (producers), thereby improving overall energy balance and reducing total execution cost in the cloud data center. and can be represented as follows:

$$X_i(t+1) = X_i(t) + k \cdot (X_i(t) - X_1(t)) \tag{4}$$

where *k* refers the factor of consumption, which is calculated based on the levy flight as follows:

$$k = \frac{1}{2}\frac{u}{v}, u \in Norm(0,1), v \in Norm(0,1) \tag{5}$$

This mechanism allows high-energy virtual machines (carnivores) to refine their tasks by referencing other efficient nodes, thereby enhancing energy efficiency. Where *Norm*(0, 1) demonstrate to the normal distribution with unit variance =1 and mean =0. The position of a carnivore is updated using a random consumer with high energy, with index (l). It can be formulated as follows:

$$X_i(t+1) = X_i(t) + k \cdot (X_i(t) - X_1(t)) \tag{6}$$

where

$$l = rand_i([2_i - 1]), i = 3, \ldots, N \tag{7}$$

Tasks are automatically reallocated to energy-efficient VMs, improving convergence and optimal selection of energy savings and execution. The position of the carnivore is updated by the position of the producer and consumer, who are randomly identified and chosen based on (l), which has the highest energy. It can also be formulated as follows.

$$X_i(t+1) = X_i(t) + k \cdot (rand_3 \cdot (X_i(t) - X_1(t)) + (1 - rand_3) \cdot X_i(t) - X_l(t) \tag{8}$$

$$l = rand_i([2_i - 1]), i = 3, \ldots, N \tag{9}$$

When redistributing tasks across cloud computing to reduce energy consumption and execution time, assign tasks to resources and achieve convergence between them. This stage is called the decomposition stage, and it is the last stage in AEO, as every worker dies, and after that, the decomposer destroys the remains. It is called the exploitation stage and can be represented as follows:

$$X_i(t+1) = X_i(t) + D \cdot (e \cdot X_b(b) - h \cdot X_i(t), i = 1, \ldots, N \tag{10}$$

where

$$D = 3u, u \in N(0,1) \tag{11}$$

$$e = rand_4 \cdot rand_i([12]) - 1 \tag{12}$$

$$h = 2 \cdot rand_4 - 1 \tag{13}$$

4 Proposed Swarm Intelligence for Task Scheduling

In this work, we address the energy and cost issues and determine the optimal path to schedule different tasks from users in a fog computing environment based on the modified AEO algorithm, utilizing SSA, and incorporating various objectives. In the proposed strategy, EAEOSSA, alternatively, solutions are enhanced by exploitation using the SSA or AEO algorithm. In the proposed strategy, EAEOSSA identifies solutions to address the problems of energy and cost associated with different tasks of users and resource optimization in cloud computing. Updating solutions through the exploitation and exploration stages can be achieved through the proposed strategy. When the exploration phase is run, the AEO algorithm is executed. When the exploitation phase is run, the SSA or AEO algorithms are run alternately to reach the optimal solutions. The following are the stages of the objectives that were integrated with the proposed strategy.

4.1 Cost of Task Schedule

The cost of scheduling tasks is crucial to sending users different tasks via cloud computing and conserving different resources. Users' main goal is to conserve various resources, reduce cost and execution time, and reduce waiting time to perform various tasks. The cost can be calculated as follows [14]:

$$Cost = \sum_{i}^{n} E_i * P_i \tag{14}$$

E_i Execution time

P_i price

4.2 Execution Time of Tasks

Calculating the length of task execution varies from one task to another, depending on user requests and their completion across nodes in cloud computing. The user aims to reduce the length of task execution to conserve resources and encourage cloud computing. The equation can also be represented as follows [15]:

$$E_{ik} = \sum_{i=1}^{n} \cdot \sum_{k=1}^{m} E_{ik} X_{ik} \tag{15}$$

E_{ik} represented Execution time

X_{ik} Boolean number of tasks execute in VMs.

4.3 Knapsack Problem

The application of the knapsack problem is an NP-hard aims to calculate the least cost and energy between nodes in fog computing. The knapsack problem contains two parameter values and weight according to the proposed strategy. When users perform different tasks, we find that the tasks take a long time to execute and wait for the users. This represents a problem in task scheduling, cost and energy consumption but this problem was solved using the knapsack problem to obtain the optimal solution. The equation can be represented as follows:

$$Max \sum_{i=1}^{m} f_i x_i \tag{16}$$

St.

$$\sum_{i=1}^{m} w_i x_i \geq v_i \tag{17}$$

$x_i \in \{0,1\}, i = 1, 2, \ldots, m$

$f_i = profit\ of\ utilization\ resource$

$w_i = weight\ of\ task\ execution\ in\ VMs$

$i = 1, 2, \ldots, m$

Each object $i \in i$ has profit r_i and weight w_i in matrix $i\ (1 \leq i \leq m)$

Binary variable $x_j = (x_j = 1)$ or not $\left(x_j = 0\right)$.

For example:

The integration of the Enhanced Artificial Ecosystem-Based Optimization with Sparrow Search Algorithm (EAEOSSA) and the Knapsack Problem (KP) aims to evaluate the algorithm's performance in solving discrete optimization problems characterized by resource constraints. The Knapsack Problem provides an ideal testbed, as it requires maximizing total profit while maintaining the overall weight within a fixed capacity. When a set of tasks is executed on a group of computing nodes, they have a specific capacity and energy. Node resources, such as power and space, are considered, and the tasks assigned to each node are based on the profit, size, and storage capacity of that node within the geographic area. The goal is to reduce response time for users, improve resource utilization, and reduce costs. Tasks are prioritized between high-value and low-energy nodes to achieve the best balance between performance and energy within the computing environment.

4.4 Makspane

The problem facing scheduling is the makespan, and the user wants to reduce the makespan in order to execute the tasks quickly. The makespan is the maximum execution of all scheduling tasks on virtual machines via fog computing. The equation can be represented as follows [18]:

$$MS = Max\left(E_j\right) 1 \le j \le m \tag{18}$$

where E_j refers to the execution time of VM, and it is calculated based on U_{ij} [19].

$$U_{ij} = \begin{cases} 1, & if\ T_i\ is\ assigned\ to\ V_j \\ 0 & if\ T_i\ is\ not\ assigned\ to\ V_j \end{cases} \tag{19}$$

$$E_j = \sum_{i=1}^{n} U_{ij} * TC_{ij} \tag{20}$$

where TC_{ij} refers the task completion time in VM, and it is calculated based on Equation [21]:

$$TC_{ij} = \frac{L_i}{PE_j} \tag{21}$$

where L_i indicates the length of the task.

PE_j indicates the processing capability of the VM.

4.5 Salp Swarm Algorithm

SSA [31] has been used recently and is inspired by the swarm of SSA in the oceans. It is also among the metaheuristic environments and its simulation of nature. SSA consists of two main parts: leaders and followers. The leader always does the best physical fitness, and the rest of the flock follows him. The algorithm calculates the suitability value to obtain the optimal and appropriate solutions for the problem. The best solution is obtained, which is the leader throughout the proposed search space. The equations can be represented as follows:

(1) Leader Phase: The leader location is updated using the following equation:

$$X_j^1 = \begin{cases} X_{bj} + c_1\left(\left(ub_j - lb_j\right)c_2 + l\right) & if\ c_3 > 0.5 \\ X_{bj} - c_1\left(\left(ub_j - lb_j\right)c_2 + l\right) & otherwish \end{cases} \tag{22}$$

c_1 decreases through the iterations as

where

$$c_1 = 2e^{-\left(\frac{4t}{T}\right)^2} \tag{23}$$

$X_j^1\ and\ X_{bj}\ repesent\ new\ placement$

c_2 and c_3 random variable from 0 to 1

$ub_j\ and\ lb_j\ refer\ the\ domin\ of\ serarch\ at\ dimintion\ j$

(2) Followers Phase: To modernize the followers' locations,

Newton's law of motion is used, which defined as

$$X_j^i = \frac{1}{2}gt^2 + \omega_0 t, i \ge 2 \tag{24}$$

So, modernizing the procedure of followers can be formulated as

$$X_j^i = \frac{1}{2}\left(X_j^i + X_j^{i-1}\right) \tag{25}$$

t *iteration*

$\omega_0 = 0$ and g velocity and the acceleration

4.6 Energy Consumption (EC)

Rationalizing energy consumption in cloud computing is crucial to avoid energy loss, cost, and waiting time for users. Using active and idle servers, VM, and DC power consumption via fog computing. Energy consumption is calculated through the following equation [26]:

$$EC = \sum_{j=1}^{m}\left(\left[E_j * \alpha_j + \left(MS - E_j\right)\beta_j\right]\right) * PE_j \tag{26}$$

α_j refer joules of VM in the active state.

β_j refer joules of VM in the idle state

$\left(MS - E_j\right)$ refer the total of time that will remain idle of VM.

Algorithm 1 demonstrates the integration of the AEO algorithm with the SSA algorithm to reduce energy consumption during task dispatch to virtual machines in cloud computing.

Algorithm 1: The proposed algorithm EAEOSSA

Input: Tasks, energy, cost, Number of tasks
Output: Optimal resources,cost and energy.
Begin
Initialize no. of tasks, energy efficient, cost.
Repeat
Randomly initialize an ecosystem
Set the fitness function
While the stop criterion is not satisfied, do
  if I = 1 then
    Production stage of AEO Eqs. (1)–(3).
  Else
    Generate rand and update its solution
  If rand < 1/3, then update its solution using
    Using Herbivore stage of AEO Eq. (5).
  Else If 1/3 rand 2/3 then
    Using Carnivore stage of AEO Eq. (6).
  If rand ≤ 1/3 then
    Using Omnivore stage of AEO Eq. (8).
  I = I + 1
Decomposition stage of AEO Eq. (10)
Else
Apply Salp Swarm Algorithm to AEO

(Continued)

Algorithm 1 (continued)

Apply Knapsack problem
Apply energy efficiency.
End while
End if
Return the optimal minimum cost and energy.

Fig. 2 illustrates the process of combining the AEO algorithm with the SSA algorithm to reduce power consumption when sending tasks to virtual machines using the knapsack problem via cloud computing.

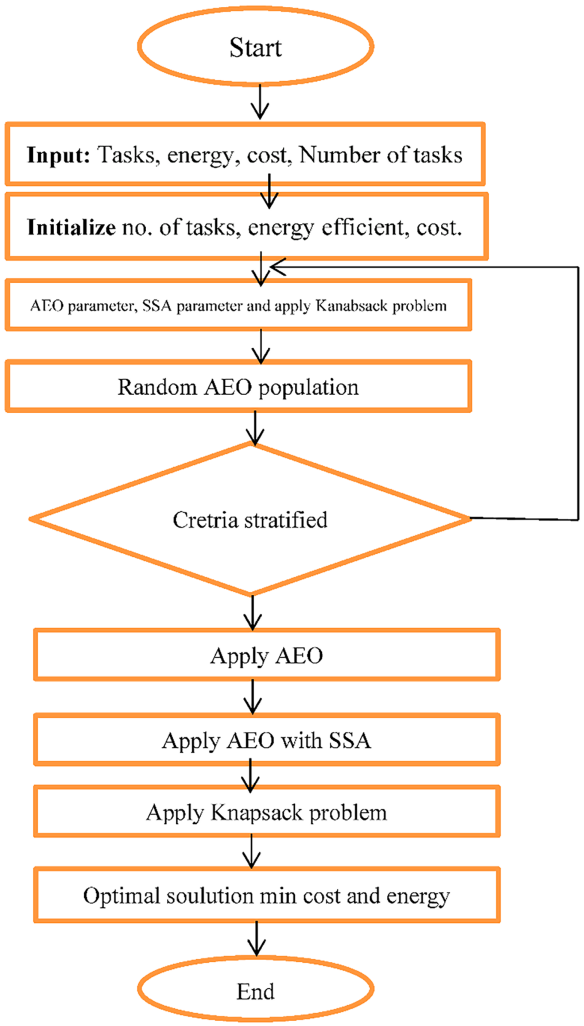


Figure 2: Flowchart EAEOSSA

The proposed task scheduling mechanism in the fog computing environment is established upon a modified AEO integrated with the SSA, defined as EAEOSSA. The proposed strategy combines the AEO algorithm with SSA to maximize the benefits of both algorithms through the exploitation and exploration phases. The strategy incorporates a competitive phase between exploration and exploitation to avoid local searches. The exploration phase is based on the AEO algorithm due to the diversity of potential solutions.

The exploitation phase, meanwhile, competes between AEO and SSA to intensify the search across global regions required by the proposed algorithm, EAEOSSA.

Complexity Analysis:

The computational complexity of the proposed EAEOSSA scheduling algorithm is expressed in terms of four main variables: the population size (N), the maximum number of iterations (T), the Number of VMs (V) and dimension (D). The hybrid optimization stage that combines Enhanced Aquila Optimization (EAO) and Salp Swarm Algorithm (SSA) involves iterative updates of the population and fitness evaluations, resulting in a complexity of O (T × N × D × V). The proposed EAEOSSA has $O(N)$.

This indicates that the proposed EAEOSSA algorithm maintains a computational complexity on the same order as other population-based metaheuristics while incorporating additional adaptive and energy-efficient enhancements.

5 Experimental Evaluation

5.1 Configuration Details

The main parameters in this paper are given in Table 2, which are used in the proposed strategy. The parameters of the AEO and SSA algorithms are also adjusted to implement the proposed strategy. The following table shows the parameters as follows:

Table 2: Parameter data sets of the system

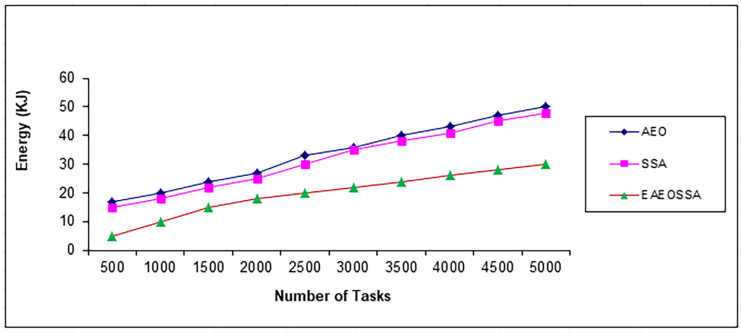
Item	Value
Nodes	[20, 100]
User	[10, 500]
Regions	[2, 30]
Bandwidth	[2, 32 Mbps]
Data sets	[2, 64 G]
Host	[100, 1000]
Processor	[12, 28]
Million Instructions Per Second (MIPS)	[500, 4000]
RAM	[2, 20 G]
Virtual machine	[50, 1000]
AEO	
Population	[50 to 100]
rand1 to rand4	[0, 1]
α	[0.5, 2]
β	[0, 1]
SSA	
Population	[50 to 100]
C1 to C3	[0, 1]

5.2 Results and Discussion

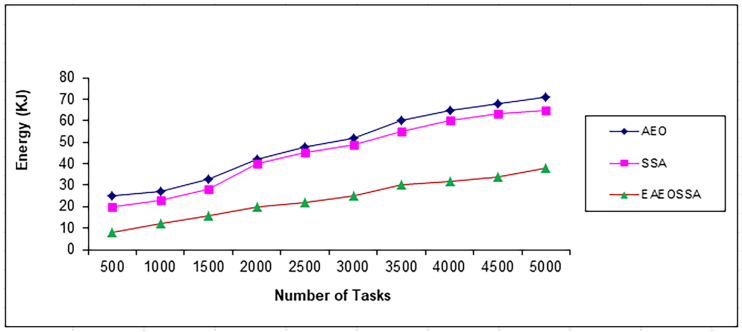
5.2.1 Energy Consumption

Fig. 3 shows the average energy consumption across different tasks from users. As the number of tasks users perform increases, the energy consumed on the various virtual machines also increases. In this strategy,

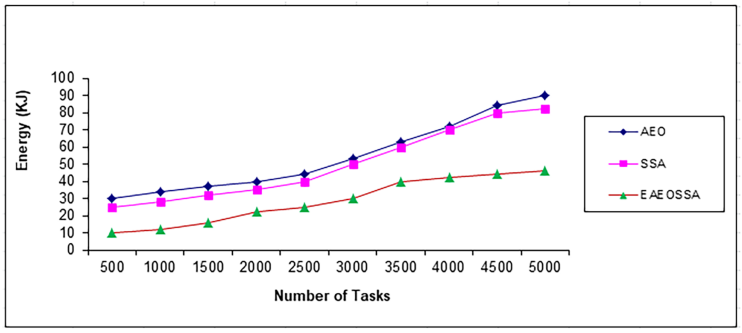
various tasks were performed, from 1000 tasks to 5000 tasks, to observe the effect of energy consumption and the efficiency of the proposed strategy. Various tasks were also applied to several virtual machines ranging from 50 to 200 VMs to test and evaluate the proposed strategy. In Fig. 3a–d, energy consumption increases very quickly when the number of different VMs increases. At the same time, the proposed strategy saves energy with different tasks sent by users. We note that the proposed strategy outperforms other algorithms in terms of saving energy for a different number of tasks and also a different number of virtual machines within the nodes in cloud computing.



(a)

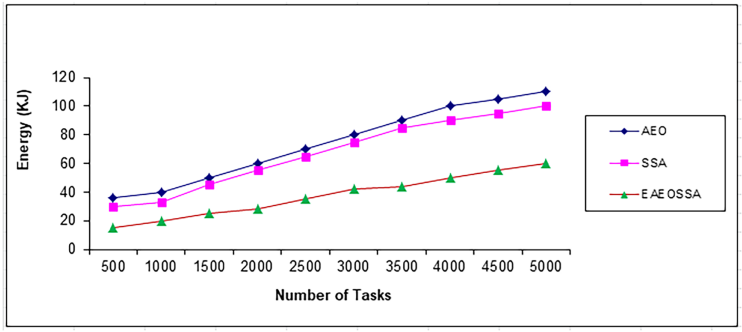


(b)



(c)

Figure 3: (Continued)

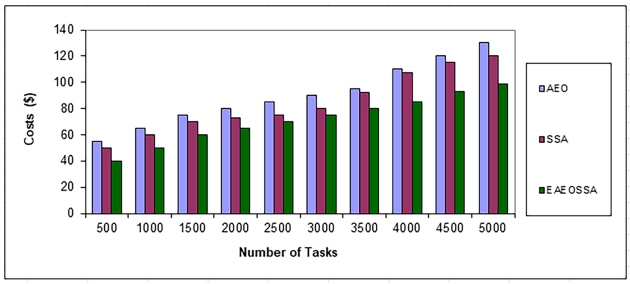


(d)

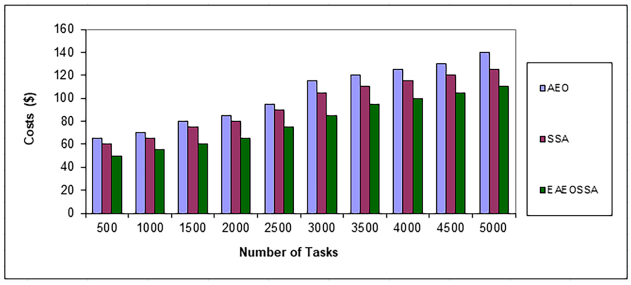
Figure 3: The impact of different cloudlets based on energy consumption using different VMs. (**a**) Impact energy awareness with 50 VMs. (**b**) Impact energy aware with 100 VMs. (**c**) Impact energy awareness with 150 VMs. (**d**) Impact energy awareness with 200 VMs

5.2.2 Cost of Tasks

Fig. 4 shows the proposed strategic cost of executing tasks on different nodes within cloud computing. As the number of tasks changes, ranging from 1000 to 5000, the cost and its impact on the proposed strategy change. It is natural that when the number of tasks increases on nodes located within different geographical locations, the cost increases. In contrast, the proposed strategy increases very slowly compared to other algorithms. Various tasks were also applied to several virtual machines ranging from 50 to 200 VMs to test and evaluate the proposed strategy. In Fig. 4a–d, when the number of different VMs increases, the cost increases, while the proposed strategy reduces the cost for different user tasks. The number of tasks from users, the task length, the number of different virtual machines and their locations in different geographical regions to execute the task, and the actual time to complete the task in nodes within cloud computing were considered. We note that the proposed strategy outperformed other algorithms in reducing time and cost for a different number of tasks and a Different Number of virtual machines within nodes in cloud computing.



(a)



(b)

Figure 4: (Continued)

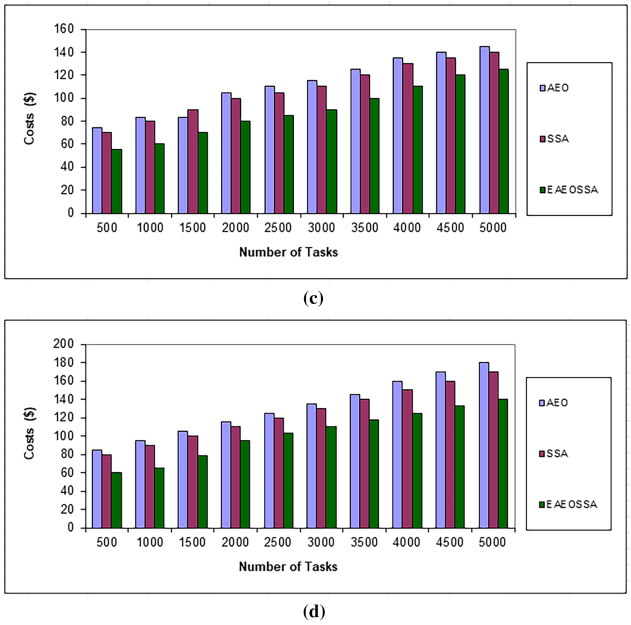


Figure 4: The impact of different cloudlets based on the cost of using different VMs. (**a**) Impact cost with 50 VMs. (**b**) Impact cost with 100 VMs. (**c**) Impact cost with 150 VMs. (**d**) Impact cost with 200 VMs

5.3 Performance Evaluation

5.3.1 Mean Service Time

Fig. 5 shows the average service time for each user task and the minimum average waiting time for users to execute tasks. In performing several tasks ranging from 1000 to 5000 in specific periods. The proposed algorithm reduced the average service time to the lowest user waiting time. It is natural that when the number of tasks increases on nodes located in different geographical locations, the average service time increases. However, with the proposed strategy, it is less than that of other algorithms. Different tasks were also applied to several virtual machines, ranging from 50 to 200 VMs, to test the average service time with different tasks sent by users over the network.

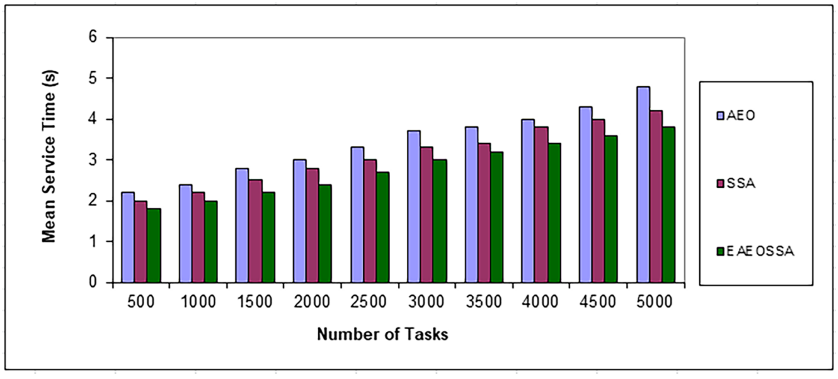


Figure 5: The impact of different response times using different tasks

5.3.2 Degree of Trouble

Fig. 6 shows the degree of imbalance between virtual machines within nodes across the network in cloud computing. In performing several tasks ranging from 1000 to 5000 in specific periods. The proposed algorithm reduced the degree of imbalance to the lowest level. It is natural that when the number of tasks increases on nodes located in different geographical locations, the imbalance also increases. However, with the proposed strategy, it decreases compared to other algorithms. Different tasks were also applied to several virtual machines, ranging from 50 to 200 VMs, to test the degree of imbalance caused by the different tasks sent by users over the network. It can also be evaluated through the following equation [27]:

$$DI = \frac{L_t}{PE_n * PE_m} \tag{27}$$

DI = Degree of Imbalance.

L_t = Total length of tasks.

PE_n = number of processing elements.

PE_m = MIPS of processing element.

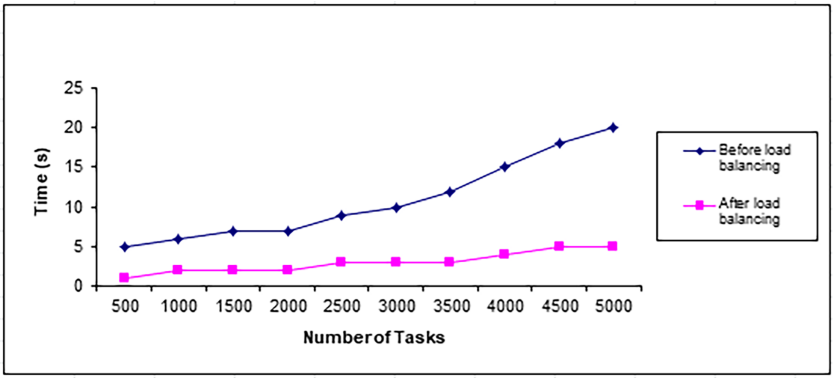


Figure 6: Degree of imbalance

6 Conclusion and Future Work

Cloud computing is one of the most recently used and reliable data transfer and storage technologies. With the increasing number of users and the advantages offered by cloud computing providers, a scheduling problem arises, manifesting in data transfer, storage, cost, energy consumption, and response time. In this research, an ecosystem-based Artificial Optimization (AEO) is developed for task scheduling via fog computing to enhance and optimize energy and cost optimization. Additionally, SSA addresses multi-objective task scheduling to enhance the AEO algorithm. The proposed strategy addresses the problem of scheduling tasks on existing resources with high efficiency and effectiveness in terms of time, cost, energy, and workload balance in fog computing. Moreover, the results of EAEOSSA were compared with those of different algorithms to test the experimental results. The experimental results showed that the proposed strategy, EAEOSSA, outperforms all other algorithms in terms of time, cost, and energy. In future work, scheduling tasks will be developed by incorporating other algorithms to enhance energy efficiency, reduce costs, and minimize time. Data availability, reliability, dependability, confidentiality, and security are also taken into consideration.

Acknowledgement: This work was supported and funded by the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University.

Funding Statement: This work was supported and funded by the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University (IMSIU) (grant number IMSIU-DDRSP2503).

Author Contributions: Ahmed Awad Mohamed, Eslam Abdelhakim Seyam, Ahmed R. Elsaeed, Laith Abualigah, Aseel Smerat, Ahmed M. AbdelMouty, Hosam E. Refaat: wrote the manuscript. Ahmed Awad Mohamed, Eslam Abdelhakim Seyam, Ahmed R. Elsaeed, Laith Abualigah, Aseel Smerat, Ahmed M. AbdelMouty, Hosam E. Refaat: reviewed and supervised. Ahmed Awad Mohamed, Eslam Abdelhakim Seyam, Ahmed R. Elsaeed, Laith Abualigah, Aseel Smerat, Ahmed M. AbdelMouty, Hosam E. Refaat: plotted. Ahmed Awad Mohamed, Laith Abualigah, Ahmed M. AbdelMouty, Hosam E. Refaat: software and methods. Ahmed Awad Mohamed, Eslam Abdelhakim Seyam, Ahmed R. Elsaeed, Laith Abualigah, Aseel Smerat, Ahmed M. AbdelMouty, Hosam E. Refaat: data analysis. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The datasets generated and/or analyzed during the current study are available from the corresponding author on reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Bittencourt LF, Goldman A, Madeira ERM, da Fonseca NLS, Sakellariou R. Scheduling in distributed systems: a cloud computing perspective. Comput Sci Rev. 2018;30:31–54. doi:10.1016/j.cosrev.2018.08.002.
2. Alkhanak EN, Lee SP, Rezaei R, Parizi RM. Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: a review, classifications, and open issues. J Syst Softw. 2016;113(1):1–26. doi:10.1016/j.jss.2015.11.023.
3. Devi KL, Valli S. Multi-objective heuristics algorithm for dynamic resource scheduling in the cloud computing environment. J Supercomput. 2021;77(8):8252–80. doi:10.1007/s11227-020-03606-2.
4. Javadpour A, Abadi AMH, Rezaei S, Zomorodian M, Rostami AS. Improving load balancing for data-duplication in big data cloud computing networks. Clust Comput. 2022;25(4):2613–31. doi:10.1007/s10586-021-03312-5.
5. Javadpour A, Pinto P, Ja'fari F, Zhang W. DMAIDPS: a distributed multi-agent intrusion detection and prevention system for cloud IoT environments. Clust Comput. 2023;26(1):367–84. doi:10.1007/s10586-022-03621-3.
6. Gupta A, Namasudra S. A novel technique for accelerating live migration in cloud computing. Autom Softw Eng. 2022;29(1):34. doi:10.1007/s10515-022-00332-2.
7. Ghasempour A. Internet of Things in smart grid: architecture, applications, services, key technologies, and challenges. Inventions. 2019;4(1):22. doi:10.3390/inventions4010022.
8. Javadpour A, Wang G, Rezaei S. Resource management in a peer to peer cloud network for IoT. Wirel Pers Commun. 2020;115(3):2471–88. doi:10.1007/s11277-020-07691-7.
9. Mahmoodi Khaniabadi S, Javadpour A, Gheisari M, Zhang W, Liu Y, Sangaiah AK. An intelligent sustainable efficient transmission Internet protocol to switch between User Datagram Protocol and Transmission Control Protocol in IoT computing. Expert Syst. 2023;40(5):e13129. doi:10.1111/exsy.13129.
10. Yang X, Rahmani N. Task scheduling mechanisms in fog computing: review, trends, and perspectives. Kybernetes. 2021;50(1):22–38. doi:10.1108/k-10-2019-0666.
11. Salem R, Abdul Salam M, Abdelkader H, Awad Mohamed A. An artificial bee colony algorithm for data replication optimization in cloud environments. IEEE Access. 2020;8:51841–52. doi:10.1109/access.2019.2957436.
12. Awad A, Salem R, Abdelkader H, Salam MA. A novel intelligent approach for dynamic data replication in cloud environment. IEEE Access. 2021;9:40240–54. doi:10.1109/access.2021.3064917.

13. Awad A, Salem R, University M, Abdelkader H, University N, Salam M, et al. A swarm intelligence-based approach for dynamic data replication in a cloud environment. Int J Intell Eng Syst. 2021;14(2):271–84. doi:10.22266/ijies2021.0430.24.
14. Mohamed AA, Abdellatif AD, Alburaikan A, Khalifa HAE, Elaziz MA, Abualigah L, et al. A novel hybrid arithmetic optimization algorithm and salp swarm algorithm for data placement in cloud computing. Soft Comput. 2023;27(9):5769–80. doi:10.1007/s00500-022-07805-2.
15. Mohamed AA, Abualigah L, Alburaikan A, Khalifa HAE. AOEHO: a new hybrid data replication method in fog computing for IoT application. Sensors. 2023;23(4):2189. doi:10.3390/s23042189.
16. Sharma M, Garg R. HIGA: harmony-inspired genetic algorithm for rack-aware energy-efficient task scheduling in cloud data centers. Eng Sci Technol Int J. 2020;23(1):211–24. doi:10.1016/j.jestch.2019.03.009.
17. Mao L, Li Y, Peng G, Xu X, Lin W. A multi-resource task scheduling algorithm for energy-performance trade-offs in green clouds. Sustain Comput Inform Syst. 2018;19(4):233–41. doi:10.1016/j.suscom.2018.05.003.
18. Shukla DK, Kumar D, Kushwaha DS. WITHDRAWN: task scheduling to reduce energy consumption and makespan of cloud computing using NSGA-II. Mater Today Proc. 2021:S2214785320392257. doi:10.1016/j.matpr.2020.11.556.
19. Khorsand R, Ramezanpour M. An energy-efficient task-scheduling algorithm based on a multi-criteria decision-making method in cloud computing. Int J Commun Syst. 2020;33(9):e4379. doi:10.1002/dac.4379.
20. Alarifi A, Dubey K, Amoon M, Altameem T, El-Samie FEA, Altameem A, et al. Energy-efficient hybrid framework for green cloud computing. IEEE Access. 2020;8:115356–69. doi:10.1109/access.2020.3002184.
21. Qasim M, Sajid M. An efficient IoT task scheduling algorithm in cloud environment using modified Firefly algorithm. Int J Inf Technol. 2025;17(1):179–88. doi:10.1007/s41870-024-01758-5.
22. Javanmardi S, Nascita A, Pescapè A, Merlino G, Scarpa M. An integration perspective of security, privacy, and resource efficiency in IoT-Fog networks: a comprehensive survey. Comput Netw. 2025;270:111470. doi:10.1016/j.comnet.2025.111470.
23. Mohammad Hasani Zade B, Mansouri N, Javidi MM. SAEA: a security-aware and energy-aware task scheduling strategy by Parallel Squirrel Search Algorithm in cloud environment. Expert Syst Appl. 2021;176(1):114915. doi:10.1016/j.eswa.2021.114915.
24. Mansour RF, Alhumyani H, Abdel Khalek S, Saeed RA, Gupta D. Design of cultural emperor penguin optimizer for energy-efficient resource scheduling in green cloud computing environment. Clust Comput. 2023;26(1):575–86. doi:10.1007/s10586-022-03608-0.
25. Javadpour A, Sangaiah AK, Pinto P, Ja'fari F, Zhang W, Majed Hossein Abadi A, et al. An Energy-optimized Embedded load balancing using DVFS computing in Cloud Data centers. Comput Commun. 2023;197(3):255–66. doi:10.1016/j.comcom.2022.10.019.
26. Goyal S, Bhushan S, Kumar Y, Rana AUHS, Bhutta MR, Ijaz MF, et al. An optimized framework for energy-resource allocation in a cloud environment based on the whale optimization algorithm. Sensors. 2021;21(5):1583. doi:10.3390/s21051583.
27. Mahmood A, Khan S, Albalooshi F, Awwad N. Energy-aware real-time task scheduling in multiprocessor systems using a hybrid genetic algorithm. Electronics. 2017;6(2):40. doi:10.3390/electronics6020040.
28. Mansouri N, Ghafari R. Cost-efficient task scheduling algorithm for reducing energy consumption and makespan of cloud computing. Comput Knowl Eng. 2022;5:1–12. doi:10.22067/cke.2022.70000.1008.
29. Zandvakili A, Mansouri N, Javidi MM. Energy-aware task scheduling in cloud computing based on discrete pathfinder algorithm. Int J Eng. 2021;34:2124–36. doi:10.5829/IJE.2021.34.09C.10.
30. Zhao W, Wang L, Zhang Z. Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm. Neural Comput Appl. 2020;32(13):9383–425. doi:10.1007/s00521-019-04452-x.
31. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM. Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw. 2017;114:163–91. doi:10.1016/j.advengsoft.2017.07.002.