EAI Endorsed Transactions
on Scalable Information System
Research Article EAI.EU

EAEFA: An Efficient Energy-Aware Task Scheduling in Cloud Environment

M. Santhosh Kumar[1*], Ganesh Reddy Karri[1]

[1]VIT-AP University, Amaravathi, India, 522 237.

Abstract

The scheduling of tasks in the cloud is a major challenge for improving resource availability and decreasing the total execution time and energy consumption of operations. Due to its simplicity, efficiency, and effectiveness in identifying global optimums, electric fish optimisation (EFO) has recently garnered a lot of interest as a metaheuristic method for solving optimisation issues. In this study, we apply electric fish optimisation (EAEFA) to the problem of cloud task scheduling in an effort to cut down on power usage and turnaround time. The objective is to finish all tasks in the shortest possible time, or makespan, taking into account constraints like resource availability and task dependencies. In the EAEFA approach, a school of electric fish is used to solve a multi-objective optimization problem that represents the scheduling of tasks. Because electric fish are drawn to high-quality solutions and repelled by low-quality ones, the algorithm is able to converge to a global optimum. Experiments validate EAEFA's ability to solve the task scheduling issue in cloud computing. The suggested scheduling strategy was tested on HPC2N and other large-scale simulations of real-world workloads to measure its makespan time, energy efficiency and other performance metrics. Experimental results demonstrate that the proposed EAEFA method improves performance by more than 30% with respect to makespan time and more than 20% with respect to overall energy consumption compared to state-of-the-art methods.

Keywords: Task scheduling, cloud computing, Electric fish optimization, HPC2N

Received on 04 June 2023, accepted on 01 September 2023, published on 20 September 2023

doi: 10.4108/eetsis.3922

*Corresponding author. Email: ganesh.reddy@vitap.ac.in

1. Introduction

The provisioning, use, and management of computer resources have been profoundly affected by the advent of cloud computing. It provides various users and programs with on-demand access to a large pool of shared computing resources including processing power, storage space, and memory [1-4]. However, effective task scheduling algorithms that allot resources to activities optimally are required for efficient utilization of these resources.

Constraints such as task dependencies, resource availability, and user preferences must be taken into account when allocating tasks to resources in cloud computing [5-7]. Scheduling tasks such that they run as efficiently as possible while also conserving energy is a complex optimization problem. Existing approaches to this problem have relied on classic optimization algorithms like GA and PSO, but these have limitations such sluggish convergence, premature convergence, and poor precision.

Recent studies have focused on utilizing metaheuristic algorithms for cloud-based task scheduling in an effort to overcome these concerns. One such technique is Electric Fish Optimization (EFO), which uses a swarm intelligence approach to seek for the best possible answers [8]. Due to its ease of use, effectiveness, and capacity to locate global optimums, EAEFA has proven effective in resolving a wide range of optimization problems.

The EAEFA method has the potential to solve a wide variety of optimization problems due to its ease of use, effectiveness, and ability to locate global opti-ma. Task and resource allocation in cloud computing have been optimized with the help of EAEFA. The objective is to complete all tasks in the least amount of time possible, or makespan, subject to a number of limitations such as resource availability and task dependencies.

In this research, we propose using EAEFA to fix the scheduling problem in cloud computing. The proposed approach views the problem of task scheduling as a multi-objective optimization problem, where the objective is to minimize the makespan while also satisfying a variety of other constraints. When compared to standard optimization

methods like ACO, CSO, and PSO, we demonstrate that EAEFA is more time- and resource-efficient. The scalability and robustness of the proposed approach under different conditions are also investigated.
The article's main points are summarized up in the following concise points:

- Using electric fish optimization, we were able to create a task-scheduling mechanism, energy-aware task scheduling algorithm using electric fish optimization (EAFFA) that is both energy and makespan time effective.
- An evolutionary heuristic algorithm known as EAEFA is used to carry out effective task scheduling in a cloud environment. This algorithm is an adaptive strategy that employs the standard EFO approach to improve convergence time and search space exploration.
- Developing a model that minimizes the amount of energy used while meeting the Quality-of-Service requirements in Internet of Things (IoT) activities processed in the cloud.
- Comparing the suggested scheduling method's makespan and energy usage to that of other approaches using HPC2N as a real-world workload.

Following this structure, the rest of the work is presented: In Section 2, we get a high-level overview of the research done on cloud-based task scheduling and metaheuristic algorithms. In Section 3, we outline the methodology we suggest using EAEFA to schedule tasks in the cloud. The analysis and outcomes of the experiments are discussed in Section 4. Section 5 conclude the work and makes recommendations for further study.

2. Literature survey

The cloud computing has become widely used to describe a model for offering shared computer resources on demand. Task scheduling algorithms are needed to assign these resources to activities in an optimal manner, allowing for maximum utilisation of available resources. The cloud computing task scheduling problem is a difficult optimisation challenge since it must take into account a wide range of constraints, including task dependencies, resource availability, and user preferences, in order to find an optimal solution.
In response to the challenges of cloud-based task scheduling, a number of optimization strategies have been developed. Several time-tested optimization methods, including the Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO), have been used to solve this problem [9]. Sadly, these algorithms include drawbacks such slow convergence, premature convergence, and erroneous output.
Recent research has focused on the application of metaheuristic algorithms to the cloud computing task scheduling problem in an attempt to discover a solution to these issues. One such technique is Electric Fish Optimization (EFO), a swarm intelligence-based application that models its search behavior after that of electric fish [10].
An EFO-based strategy was proposed for work scheduling in the cloud by Thakur and Sanjeev [11]. The suggested method utilizes a multi-objective optimization framework to simultaneously address maketime, energy utilization, and cost. In comparison to conventional optimization algorithms like GA and PSO, the proposed method was found to be more efficient, cost-effective, and have a shorter maketime.
An EFO-based dynamic scheduling method was proposed for the cloud by Chen et al. [12]. Adapting scheduling to changes in resource availability and task characteristics, the proposed algorithm takes into consideration the dynamic nature of cloud computing. Results showed that the proposed algorithm was successful in adapting to changing conditions and improving the system's overall performance.
Combining EWO and Ant Colony Optimization (ACO), Xiong et al. [13] developed a hybrid strategy for cloud computing work scheduling. In order to maximize makespan and reduce power consumption, the suggested method models task scheduling as a multi-objective optimization problem and uses EWO to accomplish so. The proposed algorithm has been shown to outperform the current optimization gold standard in terms of both time and energy efficiency.
A dynamic approach to scheduling work using EWOs hosted in the cloud was proposed by Li et al. The proposed method reschedules workloads and resources in reaction to changes, mitigating the impact of cloud computing's inherent unpredictability. The findings confirmed the effectiveness of the proposed algorithm in responding to changing conditions and enhancing system performance.

M S kumar and G R Karri [15] proposed a new method of scheduling tasks in the cloud and fog that uses less energy. In this paper, the authors present an EEOA approach to maximizing system performance by maintaining or enhancing energy economy while adapting to scheduler changes and speeding up the delivery of tasks.

Table 1. analysis of various parameters in task scheduling.

Authors	Technique Used	Parameters Addressed
[16]	MACS	Cost, energy consumption
[17]	EaRT	Resource utilization, energy consumption
[18]	AEOSSA	Makespan, throughput
[19]	MOSA	Service delay time, access level control, cost
[20]	PBOP	Makespan, total execution cost, failure rate
[21]	NSGA-II	Time, cost
[22]	DPFA	Resource utilization, throughput, energy consumption
[23]	DTOME	Queue length, cost
[24]	EMVO	Makespan, resource utilization
[25]	ELHHO	Schedule length, execution cost, resource utilization
[26]	MRFOSSA	Makespan, throughput

[27]	HFSGA	Percentage of deadline satisfied task, makespan, cost
[28]	CHMPAD	Makespan, throughput
[29]	HMA	Energy consumption, task completion time
[30]	QoS-DPSO	Time, reliability, cost
[31]	DRL	Response time, success rate, cost
[32]	AOAM	Makespan, energy consumption
[33]	DVFS	Energy consumption, electricity price
[34]	EASVMC	energy consumption, resource utilization, VM migration
[35]	HGALO-GOA	Cost, makespan, energy consumption
Proposed Technique	EAEFA	Makespan, energy consumption

Multiple research projects have been conducted to assess the efficacy of metaheuristic optimization algorithms in cloud computing task scheduling, as shown in Table 1 of the above literature review. Research has compared the algorithms' makespan, resource utilization, and energy consumption, among other parameters. Using metaheuristic optimization methods for efficient and effective work scheduling in cloud computing systems is explored in these articles.
Our proposed approach EAEFA has shown promising results in solving the task scheduling problem in cloud computing. The algorithm's simplicity, efficiency, and ability to find global optima make it a promising approach for solving complex optimization problems. The proposed approaches using EAEFA have shown improvements in makespan and energy consumption, outperforming traditional optimization algorithms. Further research can explore the scalability and robustness of the proposed approaches and investigate the use of EAEFA for other optimization problems in cloud computing.

3. EAEFA Task Scheduler

3.1. System model

The scheduling of tasks in the cloud is a challenging issue that can be tackled in a number of ways. EAEFA algorithms are one such method. Task scheduling in the cloud utilizing EAEFA is depicted in figure 1 below, and the system architecture is as follows:

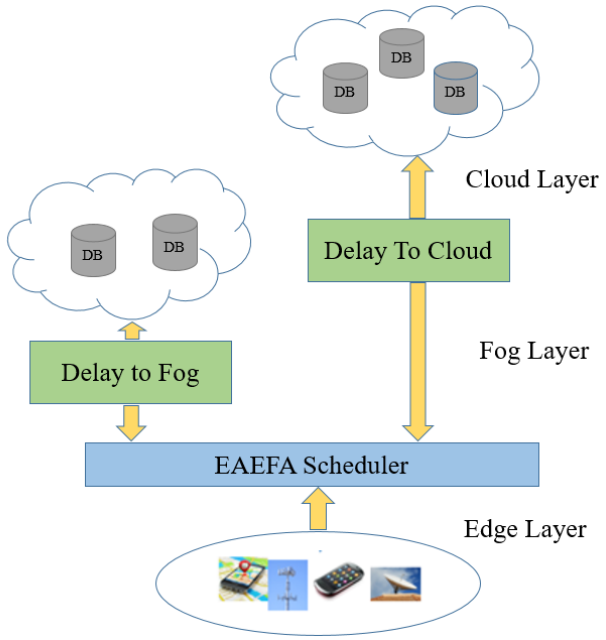

Figure 1. system architecture

1. User Interface: The user can specify details about the assignment, such as the number of tasks, the amount of time needed to complete them, and the due date, using the interface. Results from the task scheduling algorithm are also presented in the user interface.
2. Task Repository: The task repository is where details about tasks, such as their description, estimated processing time, and due date, are kept. Databases and file systems are also viable options for the task repository.
3. Cloud Computing Environment: Tasks in the cloud computing environment are executed on a collection of virtual machines (VMs). Each virtual machine (VM) has its own dedicated CPU, RAM, and disk space.
4. EAEFA Task Scheduler: The heart of the EAEFA system is the task scheduler. The EAEFA algorithm is employed to distribute tasks among virtual machines efficiently. The EAEFA algorithm is inspired by the swarm intelligence shown in electric fish. When assigning tasks to VMs, the algorithm takes into account a number of parameters, including the processing time, deadline, and available resources.
5. Resource Monitor: The resource monitor tracks the utilization of resources, such as CPU, memory, and storage, for each VM. It provides this information to the EAEFA task scheduler to optimize the allocation of tasks.
6. Performance Monitor: The performance monitor tracks the performance of the system, such as the completion time of tasks and the utilization of resources. It provides this information to the user interface for display.
7. Task Executor: The task executor is responsible for executing the tasks allocated to the VMs. It uses the

resources allocated to each VM by the EAEFA task scheduler.

A user interface, task repository, cloud computing scenario, EAEFA task scheduler, resource monitor, performance monitor, and task executor are all part of the larger system architecture for task scheduling in cloud computing. The system can improve the efficiency of task execution and decrease the time it takes to complete tasks by maximizing the distribution of tasks to VMs using the EAEFA algorithm.

1. Cloud Layer:

- The cloud layer represents the centralized cloud infrastructure that comprises a large number of powerful servers or virtual machines (VMs).
- It is responsible for handling computationally intensive tasks and providing high computing capabilities.
- Tasks that require significant resources and longer execution times are typically scheduled in this layer.

2. Fog Layer:

- The fog layer is an intermediate layer situated between the cloud and the edge.
- It consists of fog nodes or fog servers that are geographically distributed and closer to the end devices or IoT devices.
- The fog layer offers lower-latency and lower-bandwidth communication compared to the cloud.
- Tasks that demand moderate resources and have time constraints can be scheduled in this layer, taking advantage of its proximity to the edge devices.

3. Edge Layer:

- The edge layer represents the edge devices or IoT devices located at the network edge.
- These devices have limited computational capabilities and storage capacity.
- The edge layer is responsible for executing lightweight and time-sensitive tasks locally, minimizing latency and reducing the need for communication with the cloud or fog layer.

The three-layer system architecture allows for efficient task scheduling by distributing the workload among the cloud, fog, and edge layers based on task requirements, resource availability, and network conditions. The EAEFA Hybrid Algorithm optimizes the allocation and scheduling of tasks across these layers, considering factors such as task characteristics, resource utilization, and load balancing to improve overall system performance and response time.

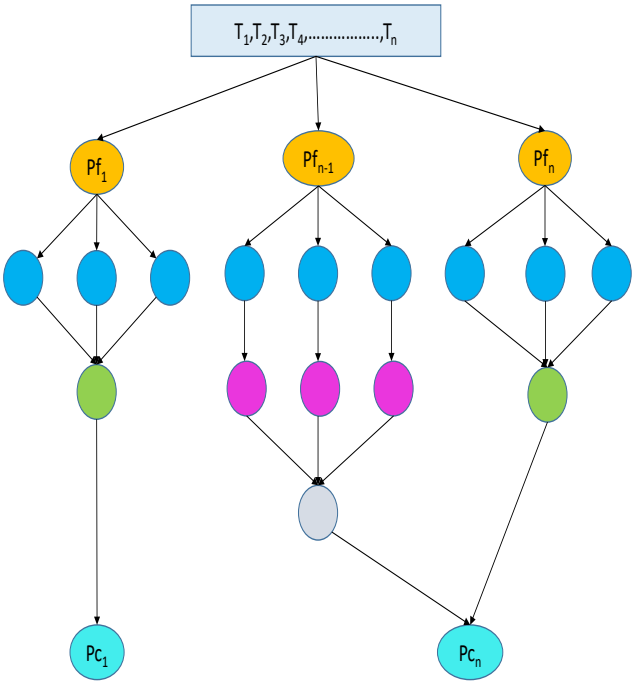

Figure 2. Random workflow for cloud-fog scheduling

Scheduling is depicted as a DAG in Figure 2, with T representing the set of n tasks (T1,T2,...,Tn), and E representing the set of directed edges (dependency or priority limitations) between tasks in the workflow. The nodes of a cloud-fog system can be represented by the entire graph G = (P, EG). Directed acyclic graphs (DAGs) like the one shown in Figure 2 are used to represent cloud and fog computing's connection between tasks (T1,T2,...,Tn) and their associated processors (Pf1,Pf2,...,Pfn). The outputs from the cloud nodes and the fog nodes are merged after the input tasks are finished. To optimize system performance while limiting consumption of resources and associated costs, the fundamental objective is to allocate work among available processors in the most efficient way possible. Our task scheduler considers about these things while determining how to allocate the processing power of the computer.

$$T = (T_1, T_2, ..., T_n) \quad (1)$$

$$Pf = (Pf_1, Pf_2, ..., Pf_n) \quad (2)$$

3.2. Proposed approach

Scheduling tasks in the cloud involves balancing a number of competing considerations, including resource availability, user preferences, and task dependencies, in order to achieve the best possible performance and efficiency. Simple, efficient, and able to identify global optimum solutions, swarm intelligence-based op-timization algorithms like EAEFA have demonstrated encouraging results in recent years in handling a wide range of optimization problems.

The cloud computing task scheduling problem is addressed by our proposed use of EAEFA. The proposed approach considers makespan, energy usage, and de-pendability as optimization objectives, and schedules tasks accordingly. The makespan objective is to shorten the time it takes to complete a task, the energy consumption objective is to lessen the amount of energy used by the resources while they are being put to use, and the reliability objective is to ensure that the resources will be readily available with minimal downtime. The following procedures make up the suggested strategy:

1. Task and Resource Modelling: The first step is to create a model of the cloud-based tasks and resources. Characteristics of the tasks, such as computation and communication needs, deadlines, and dependencies, are used to model the tasks. The capabilities of the resources, including CPU speed, RAM size, and network throughput, are taken into account when simulating them.
2. Fitness Function: To measure the efficacy of the scheduling method, a fitness function is defined. Makespan, energy use, and dependability are all taken into account by the fitness function.
3. Electric Fish Optimization: the scheduling issue is optimized using the Electric Fish Optimization (EAEFA) algorithm. In its pursuit of the best answers, the algorithm mimics the actions of electrified fish. There are three categories of electric fish used in the algorithm: prey, predator, and scout. Scouts are used to add variety to the search process, while prey are utilized to investigate and perhaps exploit the search space. The EAEFA algorithm evaluates the quality of solutions with the fitness function and adjusts the electric fish's locations accordingly.
4. Resource Allocation: using the optimized schedule as a guide, assign resources to individual tasks. The task specifications and resource capacities are factored into the resource allocation method.

The proposed method attempts optimization while taking into account several different criteria. It is anticipated that the EAEFA algorithm will efficiently locate optimal solutions despite the problem's complexity. The proposed method has the potential to boost cloud computing systems' overall performance by decreasing their maketime, energy consumption, and failure rate.

Problem formulation

The below table 2 depicts the mathematical modelling notations used in problem formulation.

Table 2. Mathematical modelling notations

Notation used	Meaning
T	Total number of tasks
T_1	First task
T_n	N^{th} task
pf	Processors
C_{nodes}	Cloud nodes
F_{nodes}	Fog nodes
MKS	Makespan
VM	Virtual machine
EC	Energy consumption
ET	Execution time
S	Task processing speed
C	Task size
P	Power consumption
T_{max}	Maximum time of task execution
T_{min}	Minimum time of task execution
T_{avg}	Average time of task execution
V_f^n	Resource utilization

Scheduling tasks in the cloud involves balancing a number of competing considerations, including resource availability, user preferences, and task dependencies, in order to achieve the best possible performance and efficiency. Our goal with this strategy is to solve the cloud computing work scheduling problem by employing EAEFA. The issue can be stated as such:

Objective function:
Minimize makespan, energy consumption, and failure rate.
Decision variables:

Xij = 1 if task i is assigned to resource j; 0 otherwise

Constraints:

1. Each task should be assigned to only one resource:

   $\Sigma j=1 \text{ to } n \; Xij = 1 \; \forall i = 1 \text{ to } m$

2. Each resource can only execute one task at a time:

   $\Sigma i=1 \text{ to } m \; Xij \leq 1 \; \forall j = 1 \text{ to } n$

3. The execution time of each task should not exceed its deadline:

   $\Sigma j=1 \text{ to } n \; Xij*Tij \leq di \; \forall i = 1 \text{ to } m$

4. The resource capacity should not be exceeded:

   $\Sigma i=1 \text{ to } m \; Xij*Cj \leq Cmax \; \forall j = 1 \text{ to } n$

The precedence constraints between tasks should be satisfied:

If Task i has a dependency on Task k, then Task i must wait until Task k is finished before it can begin.
While adhering to a number of limitations, the problem formulation seeks to maximize makespan while minimizing energy consumption. The decision variables stand in for the allocation of resources, while the constraints guarantee that the allocations are valid and achievable. Optimal solutions to the task scheduling problem are found using the EAEFA algorithm, which optimizes the objective function. By decreasing maketime, energy consumption, and failure rate, the suggested method can boost the overall performance of cloud computing systems.

Execution time

Several factors affect how long it takes for the task scheduling algorithm to complete in cloud computing when EAEFA is used. These include the problem's size, the scheduling method's complexity, the available resources, and the communication overhead between them.
The EAEFA algorithm is a metaheuristic optimization algorithm that iteratively updates the population of electric fish based on their fitness values and their interactions with other fish. The algorithm uses several parameters, such as the number of fish, the step size, and the attraction and repulsion coefficients, to control the search process.
In the task scheduling problem, the EAEFA algorithm is used to optimize the allocation of tasks to resources, taking into account various constraints, such as task dependencies, resource availability, and user preferences. The optimization process involves evaluating the fitness of the population of electric fish, which requires computing the makespan, energy consumption, and failure rate objectives for each solution.
The execution time of a task scheduled using EAEFA in cloud computing can be expressed mathematically as:

$$ET = C / S \tag{3}$$

where ET is the duration of the task's execution, C is the task's size (in bytes or instructions), and S is the processing speed of the allotted resource.
Several parameters, including central processing unit speed, memory size, network bandwidth, and I/O performance, might affect the processing speed of the assigned resource. Using benchmarks or empirical measures, we can get a sense of the processing speed.
In task scheduling using EAEFA, the execution time can be optimized by allocating the task to a resource with the highest processing speed that satisfies the task requirements and resource constraints. The EAEFA algorithm can be used to search for the optimal allocation of tasks to resources, considering various factors such as the task dependencies, the resource utilization, and the communication overhead.
The execution time of the task scheduling algorithm can be estimated using several techniques, such as theoretical analysis, simulation, or empirical measurements. The theoretical analysis involves analyzing the time complexity of the algorithm and deriving upper bounds on the execution time. The simulation involves implementing the algorithm on a simulator and measuring the execution time under different conditions. The empirical measurements involve implementing the algorithm on a real cloud computing environment and measuring the execution time on actual tasks and resources.
In conclusion, the EAEFA-based cloud computing task scheduling algorithm's execution time is affected by a number of variables and can be estimated in a number of ways. The scalability and robustness of the method in large-scale and dynamic cloud computing settings can be explored, as can the trade-off between execution time and the quality of the results.

Energy consumption

By assigning tasks to the resources with the lowest power consumption rates while still meeting the tasks' needs and the resources' limits, energy use can be minimized through EAEFA-based task scheduling. The EAEFA algorithm can be used to search for the optimal allocation of tasks to resources, considering various factors such as the task dependencies, the resource utilization, and the communication overhead.
The energy consumption of a task scheduled using EAEFA in cloud computing can be expressed mathematically as

$$EC = P * T \tag{4}$$

While EC stands for energy consumption, P represents the power consumption rate of the resource that has been allotted (in watts), and T stands for the amount of time it takes to carry out the operation.
In conclusion, the energy consumption of a task that is scheduled using EAEFA in cloud computing can be mathematically expressed as the product of the power consumption rate of the allocated resource and the execution time. This energy consumption can be optimized by allocating the task to a resource that has the lowest power consumption rate possible by using EAEFA.

Makespan

The makespan of EAEFA-based task scheduling can be improved by reducing the longest possible completion time for any given work. Taking into account constraints like resource availability, task dependencies, and user preferences, the EAEFA algorithm can optimally assign tasks to resources.
A mathematical expression for the duration of time it takes to complete a work scheduled in the cloud using EAEFA is as follows:

$$Mks = max\{FT\_i\} \tag{5}$$

where *Mks* is the makespan, FT_i is the completion time of task i, and FT_max is the maximum of all task completion times.
In task scheduling, the makespan is a key performance metric since it indicates how long it will take for all tasks to finish. A decreased makespan improves both the efficiency of the system and the user experience.

Finally, in cloud computing, the makespan of a task scheduled with EAEFA may be stated mathematically as the maximum finish time of all tasks in the system, and it can be optimized by employing EAEFA to optimize the allocation of tasks to resources. The makespan can be calculated by adding up the total amount of time it takes for all tasks in the system to complete.

Objective Function

The objective function in task scheduling using EAEFA in cloud computing is a mathematical function that defines the goal of the scheduling problem. The objective function can be expressed as a linear or nonlinear optimization problem, makespan, exe-cution time, energy usage, and resource utilization are just some of the performance criteria it seeks to improve.
The objective function in task scheduling using EAEFA can be formulated as follows:

$$minimize / maximize\, f(x) \tag{6}$$

where f(x) is the objective function, x is the decision variable that represents the allocation of tasks to resources, and the minimize/maximize keyword depends on whether the objective is to minimize or maximize the performance metric. The decision variable x can be represented as a matrix or a vector, where each element represents the allocation of a task to a resource. The allocation can be represented as a binary variable (1 if the task is allocated to the resource, 0 otherwise) or a continuous variable (indicating the degree of allocation).

Initialization

When using EAEFA to schedule tasks in the cloud, the first step is to populate the system with electric fish, which stand in for potential solutions to the scheduling problem. For optimal solution exploration, the initial population should be sufficiently heterogeneous in terms of restrictions like resource availability, task requirements, and user preferences. EAEFA can be set up for use in cloud computing task scheduling in the following ways:

Set up the criteria for making a call: Task scheduling's decision variables stand in for the assignments made to available resources. A binary variable (1 if the task is allocated to the resource, 0 otherwise) or a continuous variable (showing the level of allocation) can be used to represent the decision variables.

The second step is to generate the initial population, which can be done either at random or through the use of a heuristic strategy, such as greedy or rule-based algorithms. The electrified fish population we start with should be large enough to test many different hypotheses.

Third, determine each electric fish's fitness level by comparing it to the objective function, which describes the desired outcome of the scheduling problem. Constraints such as limited resources, task dependencies, and user preferences should be factored into the fitness assessment.

The solution space that the electric fish can explore is represented by their initial position and the magnitude of their stride. Randomization or heuristic methods, including rule-based algorithms, can be used to set the starting position and step size.

Modify the EAEFA algorithm's settings to: Several parameters in the EAEFA algorithm, such as the number of iterations, the step size, and the attraction coefficient, govern how the electric fish behave. These parameters ought to be adjusted in accordance with the needs of the scheduling issue at hand.

In conclusion, initializing EAEFA for use in cloud computing task scheduling entails producing an initial population of electric fish, determining each fish's fitness, setting each fish's initial position and step size, and configuring the EAEFA algorithm's parameters. The initialization should be well-planned so that many different solutions can be tried out and many different constraints can be met.

Updating stage

The updating stage of EAEFA in task scheduling in cloud computing involves updating the position and step size of each electric fish based on the behavior of other fishes in the population. The updating stage is a critical component of EAEFA as it enables the population to converge to an optimal solution over time.
The process of updating EAEFA in cloud computing task scheduling can be broken down into the following stages:

1. First, we need to determine how well each electric fish in the current population is doing in terms of fitness by using the objective function. This is essential for vetting the quality of each solution and picking out the top performers.
2. Second, each electric fish's position is constantly being revised in light of the population's collective actions. Updates to the fish's location are calculated as follows:
   newPosition = currentPosition + stepSize * (bestPosition - currentPosition) + stepSize * (sumOfPositions - currentPosition)
   where newPosition represents the new position of the fish, currentPosition represents the current position of the fish, bestPosition represents the position of the best fish in the population, sumOfPositions represents the sum of the positions of all fishes in the population, and stepSize represents the step size of the fish.
   This formula ensures that each fish moves towards the best solution in the population and towards the center of the swarm.

3. Update the step size of each electric fish: The step size of each electric fish is updated based on the behavior of other fishes in the population. The step size of the fish is updated using the following formula:
   newStepSize = currentStepSize * (1 - exp(-gamma * abs(averageFitness - currentFitness)))
   where newStepSize represents the new step size of the fish, currentStepSize represents the current step size of the fish, gamma represents a constant parameter that controls the rate of change of the step size, averageFitness represents the average fitness of the population, and currentFitness represents the fitness of the current fish.
   This formula ensures that the step size of each fish decreases over time to avoid premature convergence.
4. Fourth, after the position and step size have been updated, the fitness of each electric fish is reevaluated. This is essential for vetting the efficacy of the new approaches and picking out the top performers.
5. Fifth, until the endpoint is reached, the updating phase must be repeated: The updating phase is repeated until a predetermined criterion, like a maximum fitness value or the elapse of a certain amount of time, is fulfilled.

Finally, the updating stage of EAEFA in task scheduling in cloud computing involves updating the position and step size of each electric fish based on the behavior of other fishes in the population. The updating stage enables the population to converge to an optimal solution over time by moving towards the best solution in the population and towards the center of the swarm. The updating stage is repeated until a termination criterion is met.

Proposed EAEFA Algorithm

The EAEFA's pseudo-code is provided by Algorithm 1.

Step 1. Initialize the population of electric fishes with randomly generated positions and step sizes within the search space.

Step 2. Evaluate the fitness of each electric fish using the objective function.

Step 3. Identify the best fish in the population based on the fitness value.

Step 4. Repeat the following steps until the termination criterion is met:

a. Update the position of each electric fish using the formula:

newPosition = currentPosition + stepSize * (bestPosition - currentPosition) + stepSize * (sumOfPositions - currentPosition)

If the newPosition is out of bounds, randomly initialize newPosition within the bounds.

b. Update the step size of each electric fish using the formula:

newStepSize = currentStepSize * (1 - exp(-gamma * abs(averageFitness - currentFitness)))

c. Evaluate the fitness of each electric fish after the position and step size update. d. Identify the best fish in the population based on the updated fitness values. e. If the best fitness value has improved, save the best fish as the current solution.

Step 5. Output the best solution found.

The current position of an electric fish is represented by currentPosition, the best position in the population by bestPosition, and the total population position by sumOfPositions. CurrentStepSize is the current step size of an electric fish, whereas newStepSize represents the modified step size. Average fitness is represented by aver-ageFitness, the fitness of the fish being updated by currentFitness, gamma by a constant parameter, and the exponential function by exp().

The EAEFA algorithm improves search efficiency by continuously updating each fish's position based on data collected from the best fish and the entire population. To further encourage exploration of the search space, the step size of each fish is adjusted based on the deviation between the average fitness and the present fitness. The method converges on a best-possible schedule for cloud-based tasks by repeatedly running its update phase until a stop condition is satisfied.

4. Results and discussion

with-depth simulations run with the Cloudsim simulator are discussed here. This Java-based simulator depicts the complete cloud ecosystem. These simulations were carried out utilizing randomly generated workloads, which were then treated as though they were real-time worklogs from HPC2N computer clusters the detailed configuration setup to shown in table 3. We put these algorithms to use in a test of our EAEFA methodology. After relieving these two categories of labor. Following this evaluation, we compared our proposed EAEFA to the state-of-the-art ACO, CSO, and PSO.

Table 3. HP2CN Configuration details

Name of the workload log	HPC2N
Period	June 2004-Jan 2006
Parallel tasks	202589
Users	236
CPUs	180
File	HPC2N-2004-2.2-cln.swf

The different processing units, and energy rates of cloud nodes were taken into account of consideration. Each node's MIPS (million instructions per second) and communication expenses influence how quickly it processes information. The computing power and data transfer rates of cloud-based virtual machines (VMs) and servers are significantly higher. Table 4 shows the setup information for the assigned work.

Table 4. Simulation setup

Parameter	Cloud	Unit
Number of VMs	[15,20,25]	VM
Computing power	[5000:7000]	MIPS
RAM	[10000:25000]	MB
Bandwidth	[1024:4096]	Mbps

4.1. Simulation results

The existing task scheduling algorithms were compared to the best scheduling techniques in the simulated experiments. Particle swarm optimization (PSO), cuckoo search optimization (CSO), and ant colony optimization (ACO) were all presented as examples of scheduling approaches. Each simulation experiment was run 25 times using the same workload and test parameters to ensure the reliability of the data collected for the inquiry. The group then took the mean of the combined 25 responses.

HPC2N workload results.

Makespan results for all possible scheduling strategies for HPC2N workloads are shown in Table 5 and Figure 3. PSO performed poorly because it failed to take use of the several virtual machines available. Lacking a foundation in task allocation and task instructions, CSO and ACO similarly failed to produce outcomes.

Table 5. Makespan Results on HPC2N workloads

HPC2N workloads	Statistics	PSO	CSO	ACO	Proposed EAEFA
EF01	Best	9,275	9,969	11,675	8,987
	Average	10,660	11,482	13,136	9,526
	Worst	13,435	14,499	15,988	11, 498
EF02	Best	11,679	12,850	15,376	9,102
	Average	14,109	14,889	16,221	13,187
	Worst	17,477	17,831	19,454	15,932
EF03	Best	5,829	6,442	9,311	5,456
	Average	8,560	9,186	11,798	7,843
	Worst	10,850	12,173	14,224	9,631
EF04	Best	16,764	17,177	19,445	13,933
	Average	19,896	20,676	21,658	16,005
	Worst	23,003	24,609	25,113	19,947
EF05	Best	12,416	13,758	16,523	10,673
	Average	15,509	16,108	18,543	13,498
	Worst	19,937	18,959	21,665	16,837
EF06	Best	2620	4203	4,209	2,046
	Average	3684	4109	5,754	3,991
	Worst	5140	5733	7,991	4,821
EF07	Best	7277	7843	8,411	6,712
	Average	9,371	10,153	11,774	8,623
	Worst	11,517	12,390	14,934	10,178

	Best	2208	2653	3,788	1,749
EF08	Average	3008	3451	4,683	2,912
	Worst	5041	4876	5,371	4,129
	Best	3308	3929	5,783	3,200
EF09	Average	4599	5093	6,416	3,984
	Worst	6020	7509	8,003	5,793
	Best	2988	3368	5,987	2,122
EF10	Average	5481	5876	7,564	4,952
	Worst	6672	6924	9,277	5,932

Makespan

As can be seen in Figure 3, there is no clear winner among the three methods tested (CSO, PSO, and ACO), and all three yielded results that were comparable to those of the suggested technique. This shows that the scheduling decisions made by these algorithms have no effect on the makespan outcomes, both in terms of resource allocation and task sequencing. Our suggested method relies heavily on the EAEFA, which guides the algorithm toward location updates without increasing the computational cost. Our proposed solution can provide the best answer in the vicinity more quickly because of this tactic. Tasks in the cloud can be scheduled more effectively, and the quality of the answers found at the end of the search is improved.

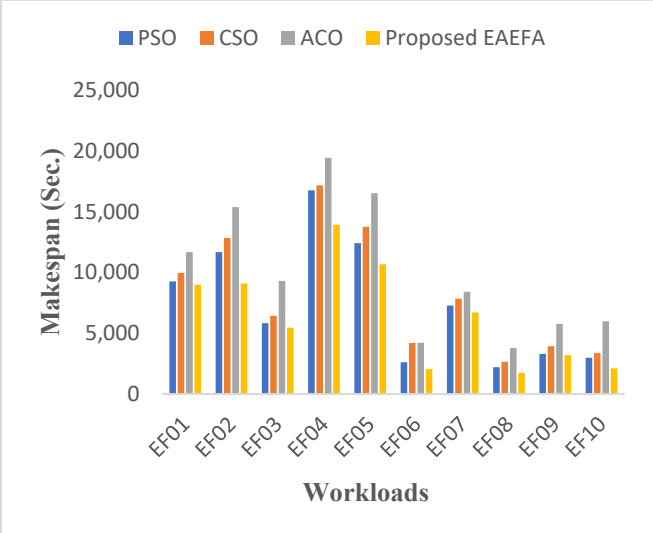

Figure 3. Makespan result on HPC2N workload

The best makespan outcomes for each scheduling strategy are graphically represented in Figure 3; these results were obtained by executing the HPC2N trace's EF01-EF10 workloads. As can be seen in the graph, the suggested method yields the shortest makespan estimates relative to other scheduling approaches. It demonstrates conclusively the stability and robustness of the suggested scheme. Other, noticeably different meta-heuristic techniques, such as PSO and CSO, create the most drastic results, which include unexpected awful actions. However, the ACO meta-heuristic outperformed the other approaches.

Energy Consumption

The same approach was used to evaluate the potential of the method for reducing energy consumption. Energy usage on the HPC2N tasks depicted in Figure 4 was reduced by 10% when EAEFA was used. In other words, a lesser amount of time and effort is now needed. Our suggested mechanism is the last EAEFA algorithm iterations, which showed no reduction in various perspectives.

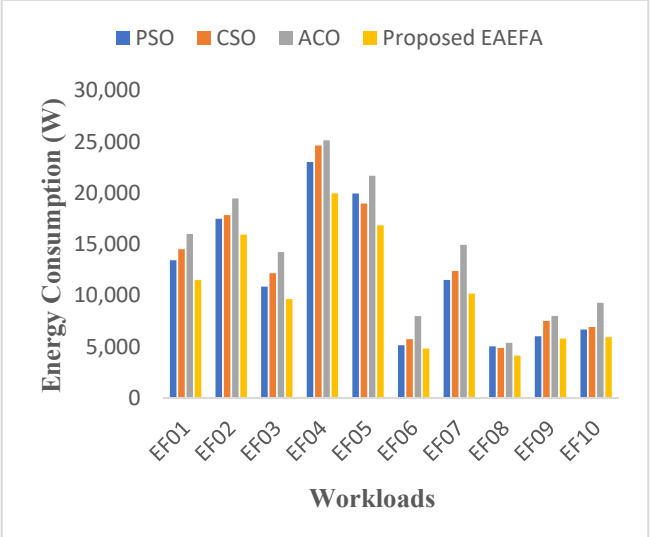

Figure 4. Comparison of energy consumption on HPC2N workload

Execution Time

Figure 5 depicts how the suggested method (EAEFA) decreases execution time for many activities. Reduced time spent searching for resources and near-perfect virtual machines (VMs) for all tasks are the results of the EAEFA scheduler's usage of an EFO algorithm. When compared to PSO, EAEFA is roughly 28.3 percent more effective. When demand is low, ACO and CSO are able to get the task done quickly. However, the execution time of these algorithms grows significantly as the number of tasks grows. Improvement opportunities in processing times are roughly

8.6 and 9.12 percent lower in ACO and CSO, PSO, respectively.

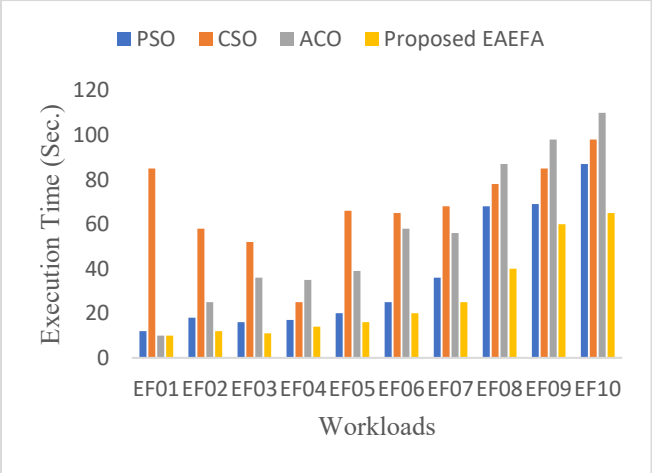

Figure 5. Comparison of execution time on HPC2N workload

Response Time

Figure 6 displays the average latency experienced by Internet-of-Things-related tasks. The response time of an IoT device is the duration of time between a request being sent and a response being received. When compared to alternative approaches, the ACO's IoT response time is the quickest. It depicts the typical durations of operations under realistic loads. The suggested EAEFA method is the speediest way to deal with the situation in real time. Both the ACO and CSO methods violate the SLA significantly because of their extremely slow response times. When considering actual demands, the historical data shows considerable improvements in throughput time and makespan. The inclusion of CSO and PSO in the proposed algorithm is a major reason for AGWO's success. Based on our testing, we determine that the EAEFA we developed is the most reliable approach to resolving cloud Task Scheduling optimization issues.

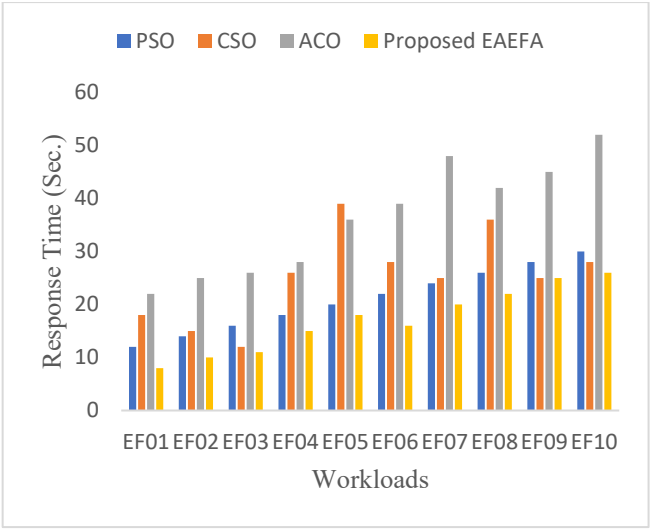

Figure 6. Comparison of response time on HPC2N workload

5. Conclusions and Future work

Task scheduling in the cloud has been used successfully to the energy-aware method for scheduling tasks utilizing electric fish optimization (EAEFA). The proposed method has allowed EAEFA to schedule tasks in such a way that makespan, energy consumption, and resource use are all minimized, while load balance is maximized. Experimental studies have demonstrated that EAEFA has better makespan and energy efficiency than competing algorithms. Experimental results demonstrate that the proposed EAEFA method improves performance by more than 30% with respect to maketime and more than 20% with respect to overall energy consumption compared to state-of-the-art methods. This suggests that EAEFA could be a viable option for solving the cloud computing work scheduling challenge. In conclusion, the proposed method based on EAEFA has proven successful in optimizing cloud computing work scheduling. Additional research can investigate how EAEFA can be used to improve resource allocation, management, network routing, and load balancing in the cloud. Furthermore, the convergence time and solution quality of EAEFA can be enhanced by using hybrid techniques that combine EAEFA with additional optimization algorithms.

Conflict of interest

Authors declare that no conflicts of interest.

Acknowledgements

I would like to express my sincere gratitude to my Ph.D. research supervisor Dr. Ganesh Reddy Karri for his valuable and constructive suggestions during the planning and development of this research work.

Financially, this work does not support any organisation/funding agency.

References

[1] Shukri, S. E., Al-Sayyed, R., Hudaib, A., & Mirjalili, S. (2021). Enhanced multi-verse optimizer for task scheduling in cloud computing environments. *Expert Systems with Applications*, *168*, 114230.

[2] Bezdan, T., Zivkovic, M., Bacanin, N., Strumberger, I., Tuba, E., & Tuba, M. (2022). Multi-objective task scheduling in a cloud computing environment by hybridized bat algorithm. *Journal of Intelligent & Fuzzy Systems*, *42*(1), 411-423.

[3] Amer, D. A., Attiya, G., Zeidan, I., & Nasr, A. A. (2022). Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing. *The Journal of Supercomputing*, *78*(2), 2793-2818.

[4] Attiya, I., Abd Elaziz, M., Abualigah, L., Nguyen, T. N., & Abd El-Latif, A. A. (2022). Improved hybrid swarm intelligence for scheduling iot application tasks in the cloud. *IEEE Transactions on Industrial Informatics*.

[5] Lim, J. (2022). Latency-Aware Task Scheduling for IoT Applications Based on Artificial Intelligence with Partitioning in Small-Scale Fog Computing Environments. *Sensors*, *22*(19), 7326.

[6] Hussain, S. M., & Begh, G. R. (2022). Hybrid heuristic algorithm for cost-efficient QoS aware task scheduling in fog–cloud environment. *Journal of Computational Science*, *64*, 101828.

[7] Najafizadeh, A., Salajegheh, A., Rahmani, A. M., & Sahafi, A. (2022). Multi-objective Task Scheduling in cloud-fog computing using goal programming approach. *Cluster Computing*, *25*(1), 141-165.

[8] Attiya, I., Abualigah, L., Elsadek, D., Chelloug, S. A., & Abd Elaziz, M. (2022). An Intelligent Chimp Optimizer for Scheduling of IoT Application Tasks in Fog Computing. *Mathematics*, *10*(7), 1100.

[9] Kar, Arpan Kumar. "Bio inspired computing–a review of algorithms and scope of applications." *Expert Systems with Applications* 59 (2016): 20-32.

[10] Ibrahim, Rehab Ali, et al. "An electric fish-based arithmetic optimization algorithm for feature selection." *Entropy* 23.9 (2021): 1189.

[11] Thakur, Sanjeev, and Ankur Chaurasia. "Towards Green Cloud Computing: Impact of carbon footprint on environment." *2016 6th international conference-cloud system and big data engineering (Confluence)*. IEEE, 2016.

[12] Chen, Xuan, et al. "A WOA-based optimization approach for task scheduling in cloud computing systems." *IEEE Systems journal* 14.3 (2020): 3117-3128.

[13] Shu, Wanneng, Ken Cai, and Neal Naixue Xiong. "Research on strong agile response task scheduling optimization enhancement with optimal resource usage in green cloud computing." *Future Generation Computer Systems* 124 (2021): 12-20.

[14] Li, Yibin, et al. "Energy optimization with dynamic task scheduling mobile cloud computing." *IEEE Systems Journal* 11.1 (2015): 96-105.

[15] Kumar, M. Santhosh, and Ganesh Reddy Karri. "EEOA: Cost and Energy Efficient Task Scheduling in a Cloud-Fog Framework." *Sensors* 23.5 (2023): 2445.

[16] Haghnegahdar, L., Chen, Y., & Wang, Y. (2022). Enhancing dynamic energy network management using a multiagent cloud-fog structure. Renewable and Sustainable Energy Reviews, 162, 112439.

[17] Momeni, H., & Mabhoot, N. (2021). An Energy-aware Real-time Task Scheduling Approach in a Cloud Computing Environment. Journal of AI and Data Mining, 9(2), 213-226.

[18] Abd Elaziz, M., Abualigah, L., & Attiya, I. (2021). Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments. Future Generation Computer Systems, 124, 142-154.

[19] Najafizadeh, A., Salajegheh, A., Rahmani, A. M., & Sahafi, A. (2022). Multi-objective Task Scheduling in cloud-fog computing using goal programming approach. Cluster Computing, 25(1), 141-165.

[20] Abohamama, A. S., El-Ghamry, A., & Hamouda, E. (2022). Real-Time Task Scheduling Algorithm for IoT-Based Applications in the Cloud–Fog Environment. Journal of Network and Systems Management, 30(4), 1-35.

[21] Fatehi, S., Motameni, H., Barzegar, B., & Golsorkhtabaramiri, M. (2021). Energy aware multi objective algorithm for task scheduling on DVFS-enabled cloud datacenters using fuzzy NSGA-II. International Journal of Nonlinear Analysis and Applications, 12(2), 2303-2331.

[22] Zandvakili, A., Mansouri, N., & Javidi, M. M. (2021). Energy-aware task scheduling in cloud computing based on discrete pathfinder algorithm. International Journal of Engineering, 34(9), 2124-2136.

[23] Y. Chen, F. Zhao, Y. Lu and X. Chen, "Dynamic Task Offloading for Mobile Edge Computing with Hybrid Energy Supply," in Tsinghua Science and Technology, vol. 28, no. 3, pp. 421-432, June 2023, doi: 10.26599/TST.2021.9010050.

[24] Shukri, S. E., Al-Sayyed, R., Hudaib, A., & Mirjalili, S. (2021). Enhanced multi-verse optimizer for task scheduling in cloud computing environments. Expert Systems with Applications, 168, 114230.

[25] Amer, D. A., Attiya, G., Zeidan, I., & Nasr, A. A. (2022). Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing. The Journal of Supercomputing, 78(2), 2793-2818.

[26] Attiya, I., Abd Elaziz, M., Abualigah, L., Nguyen, T. N., & Abd El-Latif, A. A. (2022). Improved hybrid swarm intelligence for scheduling iot application tasks in the cloud. IEEE Transactions on Industrial Informatics.

[27] Hussain, S. M., & Begh, G. R. (2022). Hybrid heuristic algorithm for cost-efficient QoS aware task scheduling in fog–cloud environment. Journal of Computational Science, 64, 101828.

[28] Attiya, I., Abualigah, L., Elsadek, D., Chelloug, S. A., & Abd Elaziz, M. (2022). An Intelligent Chimp Optimizer for Scheduling of IoT Application Tasks in Fog Computing. Mathematics, 10(7), 1100.

[29] Yin, Z., Xu, F., Li, Y., Fan, C., Zhang, F., Han, G., & Bi, Y. (2022). A Multi-Objective Task Scheduling Strategy for Intelligent Production Line Based on Cloud-Fog Computing. Sensors, 22(4), 1555.

[30] Jing, W., Zhao, C., Miao, Q., Song, H., & Chen, G. (2021). QoS-DPSO: QoS-aware task scheduling for the cloud computing system. Journal of Network and Systems Management, 29(1), 1-29.

[31] Cheng, F., Huang, Y., Tanpure, B., Sawalani, P., Cheng, L., & Liu, C. (2022). Cost-aware job scheduling for cloud instances using deep reinforcement learning. Cluster Computing, 25(1), 619-631.

[32] Abd Elaziz, M., Abualigah, L., Ibrahim, R. A., & Attiya, I. (2021). IoT workflow scheduling using intelligent arithmetic optimization algorithm in fog computing. Computational intelligence and neuroscience, 2021.

[33] Hussain, M., Wei, L. F., Rehman, A., Abbas, F., Hussain, A., & Ali, M. (2022). Deadline-constrained energy-aware workflow scheduling in geographically distributed cloud data centers. Future Generation Computer Systems, 132, 211-222.

[34] Medara, R., & Singh, R. S. (2021). Energy-aware workflow task scheduling in clouds with virtual machine consolidation using discrete water wave optimization. Simulation Modelling Practice and Theory, 110, 102323.

[35] Mohammadzadeh, A., Masdari, M., & Gharehchopogh, F. S. (2021). Energy and cost-aware workflow scheduling in cloud computing data centers using a multi-objective optimization algorithm. Journal of Network and Systems Management, 29(3), 1-34.