International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V72I5P116

Volume 72 Issue 5, 149-164, May 2024
© 2024 Seventh Sense Research Group®

Original Article

(EERO) Energy-Efficient Fog Resource Optimization Model for Scientific Workflow Applications

Satyakam Rahul[1], Vinay Bhardwaj[2]

[1]*Department of Computer Applications, Lovely Professional University, Punjab, India.*
[2]*Department of Computer Sci. & Engineering, Lovely Professional University, Punjab, India.*

[1]*Corresponding Author : satyakam_rahul@yahoo.co.in*

Received: 03 January 2024 Revised: 02 April 2024 Accepted: 15 April 2024 Published: 26 May 2024

Abstract *- Network congestion and increased latency may result from the speedy development of intelligent services and Internet of Things devices contacting cloud data centres. Fog computing meets the latency and privacy needs of operations running at the network edge by focusing on widely linked heterogeneous devices. The intricate and stringent Quality of Service limitations make allocating resources in this paradigm challenging. We look into workflow scheduling in fog-cloud systems to give an energy-efficient task plan within tolerable application completion times. The Energy Efficient optimization mode is presented. This paper delves into the outcomes of algorithms created by the researchers to address issues with energy management. The objective is to provide energy-efficient algorithms for a particular problem that minimize service compromise while reducing energy usage. The algorithms must attain a provably good performance, a crucial requirement. The goal is to find an efficient Pareto front by employing a Bayesian method with a maximum likelihood procedure for processing the fog node tasks while improving task scheduling by integrating heuristic methodologies such as Predict Earliest Finish Time (PEFT) and the Multi-objective genetic algorithm.*

Keywords *- Fog computing, Resource utilization, Energy consumption, Workflows, Energy-Efficient, Scientific.*

1. Introduction

The extensive usage of modern cloud computing has dramatically aided the direct connection between people and computer resources through networks. Communication between numerous objects, including electronic machinery, home appliances, cars, and other things, is made possible by the Internet of Things (IoT)[1]. In massive data centers, energy saving has become a top priority. Studies have indicated that data centre's yearly energy use is rising. As the Internet of Things (IoT) advances and millions of resource-constrained devices connect to the Internet in one go, billions of devices with limited resources are anticipated in future networks [2]. For now, the mobile communication network must look for a unique job scheduling method to enhance the performance of resource-constrained IoT devices. In order to fulfil the demands for service performance, a few occupations frequently developed at that time were e-healthcare, virtual reality, and autonomous vehicles. These jobs should be offloaded to a distant cloud server or other computationally heavy devices. This communication between IoT is supported by cloud computing, which generates enormous amounts of data daily [4]. The IoT and cloud data centers are separated considerably, making it challenging to handle end-user resources through the cloud. End users experience network load, mobility support, and high latency issues.

Fog technology was developed to alleviate cloud problems. Better real-time processing is made possible by the fog layer, which sits between the cloud and IoT devices. Smart agriculture, intelligent traffic management, scientific procedures, and task scheduling are examples of time-sensitive applications where fog computing is being used [6]. However, it requires efficient task scheduling while using an innovative algorithm. Numerous studies concentrate on enhancing fog computing performance while considering important issues like scheduling, privacy, security, and system deployment. The following are some of the challenges fog computing still must overcome as it develops:

- More energy consumption
- Overloading of Resources
- Privacy issues
- Security issues of Fog nodes

These factors have led to energy becoming a significant design limitation for computing equipment. Cloud companies are especially concerned about energy consumption. Based on a Google study [3], around 50% of a server's maximum power is used when it is idle. Therefore, it is essential to put servers in sleep mode whenever they are not being used to conserve energy.

The total amount of energy the computing industry uses is roughly equal to that of the aircraft sector. Its contribution to energy consumption produces 2% of anthropogenic CO_2 [7]. Numerous research efforts have been carried out to increase the energy efficiency of data centers, including bettering air conditioning, equipment, and data center design [8][9]. Two key strategies for energy conservation and green computing are increasing energy usage efficiency and lowering energy consumption. One such research issue that has the potential to provide significant results is efficient to work scheduling in data centers. Computers can use less energy to execute tasks when task scheduling is adjusted.

Additionally, it lowers energy usage from auxiliary equipment. Effective work scheduling in a big data center can result in significant energy savings. There will be a significant impact on practice if the energy consumption of multiple computations is reduced by several orders of magnitude (n). The power used by computer servers alone ranges from 23 to 31 gigawatts, accounting for $14 to 18 billion in annual costs and 1.1 to 1.5% of the world's electricity use [13]. The past several years have seen a lot of research interest in algorithmic methods for energy conservation. Energy-saving concerns are the fundamental factor for fog-based IoT networks. Due to their distributed locations, most Internet of Things devices have limited energy, making it challenging to keep their formal work continually. An energy-consuming scheduling strategy that can minimize the makespan of an application is not an appropriate solution for fog resources. This task becomes increasingly difficult when numerous conflicting objectives need to be fulfilled at the same time. Reducing both makespan and energy usage in application processing is a difficult task. Hence, a bi-objective optimization methodology is necessary to determine the optimal balance between these optimization objectives. When it comes to IoT networks, the Energy Efficiency (EE) of the network is a crucial metric to show system performance, which is the focus of this study.

This paper assesses and discusses various scheduling meta-heuristic algorithms on time, cost, and energy-consuming parameters. We suggest a distinctive algorithm, the "Energy-efficient Fog Resource Optimization Model," or EERO, to reduce energy costs while adhering to a specific timing restriction. The model is made up of three parts: clouds, fog, and edge. The reduction of overall energy costs is the aim. When there are more available heterogeneous computing units or input jobs, the complexity of the computation should increase exponentially.

With the model in place, we suggest a heuristic approach to produce near-optimal solutions that consider the overall performance of a whole system. By incorporating heuristic techniques into enhanced job scheduling, we overcame the first difficulty of minimizing energy use, such as PEFT and the Multi-objective genetic algorithm. Further time and cost of task processing are reduced through efficient task ranking. As a result of using a Bayesian method, task scheduling for workflows on Fog nodes is more efficient and reliable.

Since GA places only a few restrictions on the optimization issue, it has found extensive use in numerous fields and produced excellent outcomes. The search speed and solution quality, however, are not always sufficient. The performance of GA has been significantly improved, by adding local search, adding adaptive capability, and hybridizing with other algorithms. We add a new feature to the GA search that can lead to considerably better results. The goal of this work is to find an efficient Pareto front by employing a Bayesian method with a maximum likelihood procedure for processing the fog node tasks.

The work's primary contribution is

- Improving task scheduling using the integration of heuristic methodologies such as PEFT and the Multi-objective genetic algorithm.
- Reduce the randomness of the optimization procedure in order to create fog scheduling.
- In high-performance fog computing, which includes execution effectiveness and energy usage, this study solves a crucial issue of energy consumption. Our method allows for the creation of nearly ideal solutions.

2. Related Work

Resource management has become more challenging due to fog computing technology's ongoing development. The strategies for managing resources currently in use are presented in this section, along with their benefits and drawbacks. Under quality-of-service standards, there are many difficulties with sophisticated resource allocation and communication resources. In wireless IoT networks, the problem of job scheduling and resource allocation for several devices is being researched. A technique called Energy Harvesting (EH), which enables devices to obtain energy from the environment, was put forth by Chang et al. in 2020. The authors suggested using the Lyapunov optimization algorithm to lower the execution cost [10]. Huang et al. 2020 solved the problem of energy-efficient resource allocation in fog computing networks. They suggested a Fog Node (FN)--based resource allocation technique and transformed it into Lyapunov optimization to boost the network's energy efficiency. The scheduling algorithm used most frequently is Heterogeneous Earliest Finish Time (HEFT) [21]. The two phases in which HEFT functions are the task prioritizing and processor selection phases. The tasks are given priority in the first phase based on their ascending ranks. In the second phase, suitable processors are selected in consideration of the shortest possible job completion time. Predict Earliest Finish Time (PEFT) is a well-known method in this field [22]. The PEFT scheduling phase uses the Optimistic Cost Table (OCT), computed to prioritize tasks and help choose the best processor for task execution. HEFT and PEFT are both singleobjective optimization strategies that only consider makespan minimization, in contrast to EERO, a multi-objective optimization technique that considers energy consumption. A hybrid meta-heuristic strategy combining the Genetic Algorithm (GA) and Ant Colony Optimization (ACO) is suggested in [23] for reducing makespan in a multi-processor cloud setting. Priorities are set at the task's bottom level (b-level). The b-level is the longest possible execution time for a job to traverse the entire graph. Then, using GA, ACO is used to find a suitable path that is then further enhanced.

Communication issues resulting from the massive amount of data flows were made worse by big data. Hence, the communication issue has been addressed via fog computing. A resource management issue brought on by the quantity of available heterogeneous computing was solved via fog computing [11]. A unique load strategy and load-balancing algorithm were developed by Jamil et al. This method seeks to increase the use of fog devices in a specific area while reducing energy consumption and delay [12]. Deng et al. [13].'s depiction of the fog-cloud interaction considers the tradeoff between delay and energy consumption. Although the others are transmitted via a vast area network to the cloud, the requests that require speedy responses are handled locally on the Fog devices (WAN). A heuristic-based algorithm is proposed by Pham et al. [14] to strike a match between the financial cost and make-span of cloud resources. It establishes the job priority and selects an appropriate node for each task. The findings demonstrate that the suggested method outperforms the GfC, HEFT, and DLS algorithms regarding cost-makespan tradeoff value. The classification mining algorithm is the foundation for the Task Scheduling in Fog Computing (TSFC) algorithm [24].

The job completion durations and the association rules derived from the I-Apriori algorithm are combined without accounting for machine bandwidth. The makespan is reduced by task scheduling in software-defined embedded systems (FC-SDES) that are supported by fog computing [25]. It suggests a simple three-phase approach that combines resource management, task scheduling, and I/O request balancing. DVFS-enabled The Dynamic Voltage and Frequency Scaling (DVFS) technique is employed by the state-of-the-art Energy Efficient Workflow Task Scheduling (DEWTS) algorithm in the cloud environment to optimize energy usage during available time slots [26].

A decent solution can be found using heuristic algorithms that consider all possibilities. They frequently finish the searches exceptionally quickly. They may only sometimes find the finest solutions but always provide the locals' best ones. Typically, these algorithms can locate answers close to the ideal ones. The search space is combed using bio-inspired search algorithms that mimic natural processes. The usual algorithms are Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and others. They can locate ideal or almost ideal solutions. Modern computer architectures perform more slowly than heuristic algorithms. The goal of hybrid algorithms is to integrate the best features of several algorithms.

3. Methods

Fog Computing Architecture for Efficient Energy Consumption. Fog computing is a decentralized computing infrastructure in which data, computing, storage, and applications are located somewhere between the data source and the cloudSensors can only capture and transmit data; they cannot compute or store it. Actuators are used in conjunction with sensors to regulate the system and respond to environmental changes sensed by the sensors. A framework is needed to improve energy efficiency in fog computing, an application based on scientific workflow. We propose EERO (Energy Efficient Resource Optimization) for Fog computing to reduce the cost, execution time, and energy usage. In Figure 1, the suggested EERO model is exhibited. There are three layers of energy-efficient fog architecture for resource optimization. The following are the layers: the fog layer, the end-user layer, and the cloud layer. The suggested design has the same features, just like the fundamental architecture of fog computing, but with an additional modified fog layer. The explanation of these strata is provided below.

End-User Layer: End users generate requests at the network's edge and send them to the fog layer. Applications for scientific workflow generate millions of jobs each second due to the rising demand. Prior to being delivered to the fog layer, these tasks are processed to be carried out. For efficient work distribution, we use the Pareto distribution method. Other jobs are transmitted to the cloud layer, while these are carried out in the fog layer itself.

Fog Layer: Several clusters with a few fog nodes have been formed from the fog layer possessing fog nodes. Every cluster has a local controller that keeps track of every fog node and ensures its resources are used to their full potential. Users associated with the fog layer constantly transfer demands to fog nodes. Due to numerous users, a sizable number of tasks are also developed. This architecture will place connectivity services near the data-generating nodes at the lowest awareness layer. The system is composed of computers, physical and virtual sensors, nodes, and other elements. Nano data centers with limited cloud-like services are found in the fog layer. The processing and storage capacities of these small data centers are constrained. Therefore, only those jobs that need to be completed immediately are carried out, while others are transferred to the cloud layer.

Cloud Layer: The cloud layer is joined to the fog layer for future data transmission and archival. Large data centers with abundant networking, storage, and processing power are found in the cloud layer.

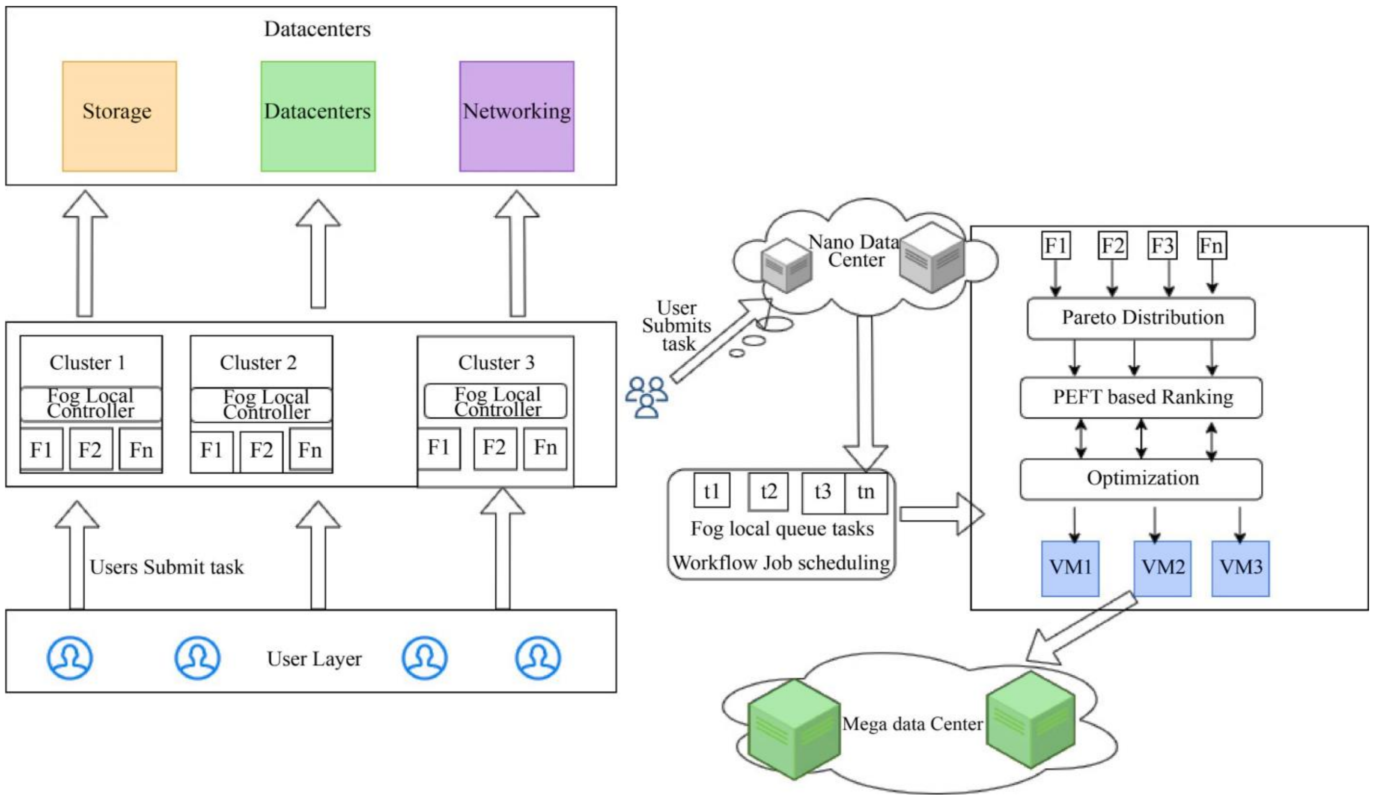


Fig. 1 EERO

These data center offers repository assistance for the nano data centers less significant priorities that will be stored and used in the future.

3.1. Operating Module of EERO

Regarding operating procedures, the suggested model is separated into three modules: an optimization module, a pre-processing module, and a parameter analysis module. According to Figure 2. Below is a description of the modified Process for each module.

3.1.1. Initial Processing or Pre-Processing Module

The Workflow Management System (WFMS) is used to split workflows into a collection of activities, facilitating the automatic and efficient execution of the workflow. It enables users to design and analyze workflows, set budgets as well as time limits, and select the working conditions they like. We apply Pareto distribution to distribute tasks efficiently and within budget and deadline. After analyzing and placing these inputs into action within the established restrictions, the WFMS After looking at the dependencies, the task dispatcher delivers to the scheduler the finished assignments.

3.1.2. Optimization Process or Module

The user is given complete knowledge of the service they received when completing several assignments using this method. If all the nodes receive the resources, the number of jobs assigned to fog nodes is completed. However, resource optimization is required when a few activities do not receive supplies or resources and the fog layer's nodes are still underloaded. We apply the PEFT ranking algorithm to the task available.

3.1.3. Analysis Module for Parameters

Analysis of the examined parameters, such as cost, energy consumption, and execution time, occurs after resource optimization. If it is discovered that optimization is still necessary while assessing the parameters, tasks are returned to the optimization module and rescheduled.

3.2. Assignment of a Workflow Task

The Process of task distribution in fog nodes is explained here. The workflow organizer compiles user-generated tasks from various users, adds all of them to a queue, and leaves them until processing resources are available. At fog nodes, distant users send their workloads for execution. According to their priority, these jobs are delegated by the workflow scheduler to the local controller of fog. Tasks are dispatched for execution when resources are available. The task scheduler is updated on the task status following task execution. When tasks are given as resources become available, the load on the fog layer can decrease this way. The coordinator node of the fog cluster monitors the load dissemination across virtual machines. Tasks are moved from the overloaded Virtual Machine to the idle Virtual Machine in question when Virtual Machines are overloaded.

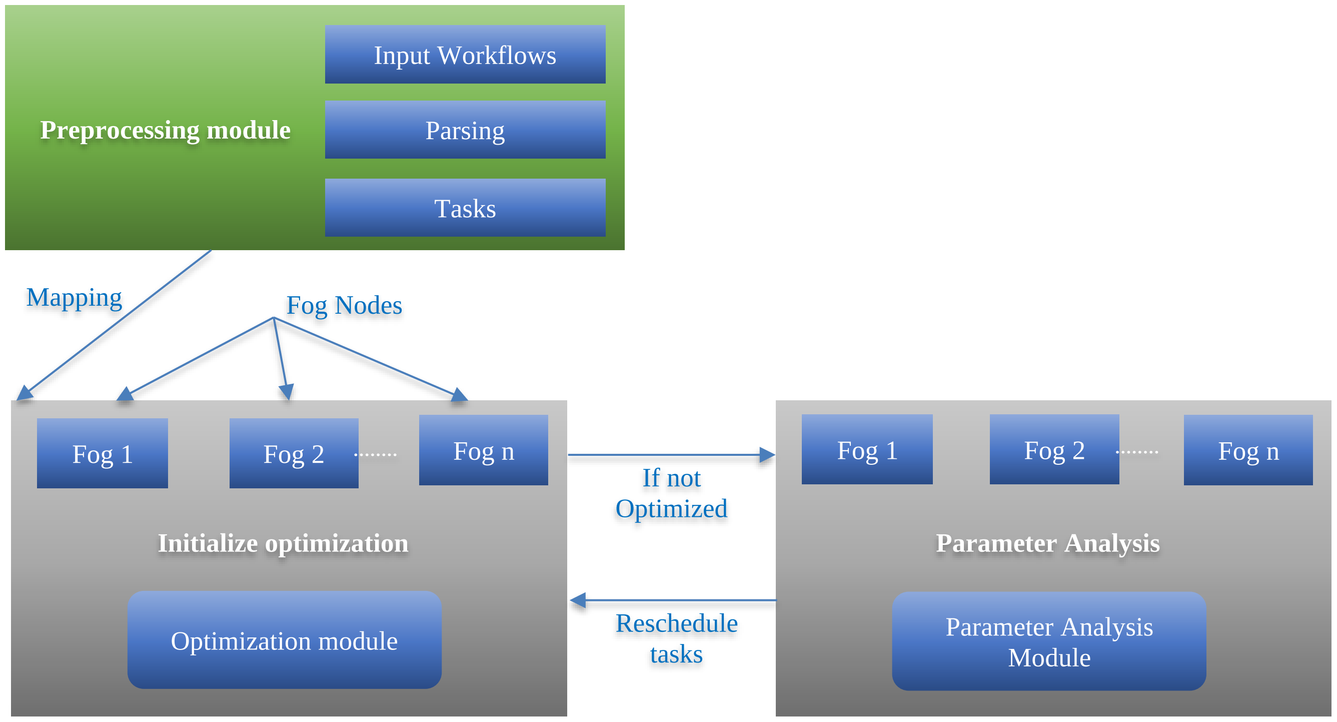


Fig 2. Workflow of EERO

3.3. Proposed Workflow Model

A collection of vertices (V1, V2, V3..., Vn) can serve as the representation of a Directed Acyclic Graph (DAG), and edges serve as the specification of a process in fog computing (E1, E2, E3..., En). Workflows can be described as NP-complete problems in fog computing. The tasks linked to the collection of VMs are thus indicated by the vertices, which are "VM1, VM2, VM3..., VMn," the edges represent the interaction between tasks T, i.e., "T1, T2, T3…, Tn." Workflow weights are applied to edges by supplying a time for computation and interaction for each job. Resources R, i.e., "R1, R2, R3…, Rn," in the fog and cloud layer, are assigned to these tasks. The time, cost, energy, makespan, and objective function models for fog computing processes are presented in this section.

3.3.1. Time

When the workflow is being executed, the numerical solution can follow the execution phase of the workflow or schedule the remaining tasks once again by measuring the available execution time. Job dependencies, task heterogeneity, and computational capacity should all be considered when determining the workflow execution period [16]. The fact that some runtime components of scientific workflows are designed stochastically from the estimation perspective is another crucial aspect of these processes. The time a process requires from capitulation to conclusion or completion is used to compute the execution time in workflows. This time considers any processes still waiting their turn, such as the time required to await supplies or resources or another task to finish.

$$T_{ti} = \sum_{x=1}^{VMx} T_{Re} + \sum_{x=1}^{VMx} Tp + \sum_{x=1}^{VMx} Tw \qquad (1)$$

T_{ti} = Total time
T_{Re} = Time required to receive a task
T_P = Time required in the processing of a task
T_w= Waiting time for a task

Where VMx indicates how many virtual machines are there overall.

3.3.2. Cost

When implementing scientific procedures, both the Cost Factor (CF) and the overall Movement Factor (MF) are taken into account (CF). The ratio of expenses incurred during task execution considering migration and Virtual Machine (VM) costs is known as MF. CF is determined as the proportion of the total cost of the Process to the Cost of the VM and the data center multiplied by the amount of memory used by the task.

$$T_{Co}\ _{(Total\ Cost)}=(MF+CF)/2 \qquad (2)$$

Where MF defines the Movement factor, and CF defines the Cost factor

$$MF= \frac{1}{Total\ number\ of\ hosts\ in\ data\ center}$$

$$\sum_{x=1}^{VMx} \frac{Number\ of\ migration}{Used\ VM} \qquad (3)$$

$$CF = \sum_{x=1}^{VMx} \frac{Cost\ to\ process * memory\ of\ tasks}{VM * Data\ Center} \qquad (4)$$

Where VMx indicates how many virtual machines are there overall in the system. It is possible to calculate the overall cost of a task that is on time and a late task.

Actual cost = Cost of Underlined tasks + Cost of tasks which have crossed the deadline (5)

3.3.3. Energy

Energy is calculated as the total of all instances' movement factor, time, and cost factor. The following equation depicts how much energy the fog environment uses when running operations.

$$\text{Energy} = \sum_{x=1}^{VMx}(T_{ti} + MF + CF) * \text{Number of instances} \quad (6)$$

The three terms Tti, MF, and CF stand for total time, movement, and cost, respectively.

3.3.4. Makespan

It is the anticipated amount of time required to calculate the matrix ETC (T_j, R_j), where T_j is a list of tasks, and R_j is a list of available resources required to finish each task. The postcondition shows that the actions should be completed in a particular order to shorten the makespan. To shorten the makespan, the load is dispersed among the resources available. The makespan (MS) is determined using the formula below.

$$\text{MS} = \text{maximum}\ (\text{C}(\ T_{j,}RR_n)) \quad (7)$$

The formula C=R R n + ER n is used to compute C, the task completion time. RR_n denotes the resource's ready time in this case. ER_n is a representation of the time that task j took to complete for resource n.

3.3.5. Objective Function

The aim of this study can be described as follows using the makespan, cost model, energy model, and time model that were determined previously.

$$f(p) = \alpha * (T_{ti} + T_{co} + E + MS)$$

Here, the model's objective function f(p), and it should be as small as feasible for the optimum outcome. When an algorithm achieves optimization, fitness value is realized, and Total Cost, energy, total time, and makespan are each represented by the letters Tc, E, Tt, and MS, respectively.

Scientific Workflow Applications

It is described as a collection of interconnected tasks represented by a Directed Acyclic Graph (DAG). The tasks' nodes are the graph's nodes, and its edges are its edges. Various sensors and actuators produce these jobs in numerous application scenarios, including astronomy, e-healthcare, intelligent traffic management, and many more [17].

Task representation in scientific workflows takes the form of a DAG. Numerous scientific workflows exist, including Genome, Cybershake, LIGO, Sipht, and Epodonomic [18]. In a DAG, tasks are specified by edges and nodes that are connected to one another. The edges and nodes reflect communication between the tasks. Optimizing scientific workflow scheduling can make fog computing much more efficient overall. Workflow benchmarks are essential for the development and evaluation of workflow management systems. This work aimed to decrease the execution time, expense, and energy consumption of a few scientific workflows that use the suggested technique to implement these workflows.

LIGO

Laser Interferometer Gravitational Observatory, or LIGO, is a term used in physics. It is used to measure the gravity of the earth. LIGO's huge number of tasks requires considerable memory and processing power [19]. Most LIGO jobs require VMs with memory optimization. Data from small binary systems, such as binary neutron stars and black holes, are interpreted using this method [20]. The observer attempts to measure and detect waves like relativity predicts. The procedure generates a subset of spatial domain output waveforms for each section and assesses the output of the corresponding filters. If a true inspiral is found, a trigger is made, which can then be compared to stimulus for all the other detectors.

Cybershake

The grid-based SCEC environment uses a workflow architecture to implement the Cybershake computational process[28]. The Southern California Earthquake Center uses Cybershake to describe the earthquake risks[29]. Cybershake is another data-intensive procedure with high CPU and memory demands. The SCEC initially used it to describe the region's earthquake risks. Given the region of interest, Strain Green Tensors (SGT) is generated using an MPI-based differential simulation. From the SGT data, synthetic seismograms are produced with each of the anticipated ruptures[30,31].

Genome

The genomic program, CPU-intensive, is used by the epigenome center to analyze the output data of DNA methylation and histone alteration results. The Illumina-Solexa Genetic Analyzer's DNA sequencing lanes are initially used to capture the data (ISGA). DNA sequences are produced in multiples by each Solexa machine. There are several sections of the ISGA. The data from each chunk is converted into a file that the Maq framework can read [32]. The procedure then maps DNA sequences to specific locations in a genome to produce a map showing the sequence density. When DNA sequences are mapped to precise locations in a genome, a map showing the density of the sequences is produced. A global map must be made; distracting sequences must be eliminated; arrangements must be translated into the proper position in a genome, and sequencing abundance must be calculated at each point [33].

Sipht

At Harvard University, Sipht is utilized in bioinformatics efforts to find bacterial reproducing. It is applied to find short bacterial RNAs (sRNA), which control how bacteria secrete. A national center automates the search for expressing and decoding genes in sRNA using the Sipht methodology [34,35]. Bioinformatics research at Harvard University sought short and untranslated RNAs (sRNA) that could regulate several bacterial functions like secretion and pathogenicity. The investigation for sRNA-encoding genes across all bacterial reproduction in the NCBI database is made more accessible by the sRNA recognition methodology, which uses technological innovation software [36].

3.3. EERO for Scientific Workflows

The section provides an overview of the various optimization methods that were employed in this study and suggests a hybrid solution for scientific operations using a novel approach

3.3.1. Optimization Method Used

Utilizing EERO optimization techniques, the main goal is to reduce energy usage. The proposed architecture is divided into four parts.

- Parsing of Workflows
- Optimize the ranking
- Optimize the task scheduling
- Analysis the parameter

Therefore, the first step is to identify the optimal Pareto front, followed by a PEFT-based ranking of that region. After ranking, probability distribution was determined correlation between these task ranks and optimized using a Bayesian strategy. If no goal can be improved without sacrificing at least one other goal, the Pareto front is a group of nondominated solutions picked as the best option. On the other hand, a solution x* is said to be dominated by another solution x only when x is as good as or better than x* in terms of all objectives.

As rankings depend on one another, we use the data from the previous steps to create an efficient task mapping. As a result, NSGA II is utilized to learn through multi-objective optimization utilizing equation 1. observe resource usage, and propose an ideal work scheduling threshold for virtual machines. The reason to use Bayesian optimization is to find the global minimum in the fewest steps. This technique offers a beautiful framework for tackling problems that mirror the scenario.

So, parse the workflows as per the parent-child relationship and the specified order, although many tasks will appear in the series. On the same level, we go to the next phase and assign an optimal ranking

Fitness function

$$\mathrm{F} = \partial\ (\mathrm{ET} + \mathrm{EC} + \mathrm{E}) + \lambda(ET + EC + E) \ldots. (1)$$

∂ =learning parameter

λ =optimize parameter

ET=execution time

EC=execution cost

E=energy

First, set N and W to the initial values for the number of fog nodes and workflows, respectively. Fog nodes are converted into parser trees when the nodes and workflows are established. Once fog node parsing trees are created, high computational workflow jobs are broken down into smaller tasks, extracted from workflows, and mapped onto fog nodes. Finding the ideal Pareto front is the first step. It provides the answer to the multi-objective optimization problem. The Pareto front provides a set of non-dominating optimal options.

The Pareto front algorithm is as follows:

```
Input: Define Objective
Output: Optimize Pareto front
1. Dß find dependency in tasks on the basis of Time and energy
2. if $t_i > t_J$
        start
            dominate ($t_i$) by eq(1) $>$ domination ($t_J$)
            Run step2
            if converge
        stop
3. Front according to converge
4. Run according to eq(1)
```

After finding the optimal Pareto front, we imply a PEFT-based ranking of that region.

```
PEFT -Ranking
            Input: Pareto front space
            Output: ranking according to Pareto front
1. D←find dependency in tasks by pareto
2. if $t_i > t_J$
        start
            dominate ($t_i$) $>$ domination ($t_J$)
            Run step2
            if converge
        stop
3. Ranking according to converge
4. Run according to objectives
```

After ranking, determine the probability distribution correlation between these task ranks and optimize using a Bayesian strategy, as rankings depend on one another.

Bayesian Optimization

```
Input: DAG (workflows) with PEFT Ranking
Output: Optimize Ranking
1. While (Task> mean μ(x|D) *N)
        start
            $T_c$ ß calculate task time and Energy in every fog node
                        $T_u$ ß compare upward
                2. if $T_u$>$T_c$>
                3.   Bayes (Optimize)ß $T_c$
                4. else
                5. Rank $_{High}$ ß $T_u$
                Stop
```

3.3.2. Performance Assessment

This section represents the findings of the simulation from iFogSim, which is a simulator for IoT, edge computing, and fog environments for IoT service management, and modeling and simulating networks and diverse applications, are presented. Likewise, iFogSim is compatible with CloudSim, which has a vast collection of tools for managing resources and simulating cloud environments. CloudSim manages the events that occur between the fog parts.

3.3.3. Criteria for Performance Assessment

This work seeks to offer an energy-efficient resource optimization for workflow-based fog computing applications. The suggested method reduces fog node execution time, implementation costs, and energy usage. The validity of the suggested work has been examined using three different types of experiments. This paper offers three test cases to demonstrate the experiments' findings.

In the initial test scenario implementation costs have been examined. Workflow execution times for applications are reflected in the second test case. The next test case represents how much energy was consumed by various resources. The calculated results have been obtained from 2 to 200 fog nodes. Around forty runs have been performed to guarantee the accuracy of the studies.

3.3.4. Experimental Setup

Several experimental requirements have been considered to evaluate the proposed technique. The conditions needed to produce simulation results are listed in Table 2. The Windows 7 64-bit operating system was employed. The reliable tool for simulation, iFogSim, has been utilized to display simulation results. Two thousand instructions are applied for execution every second in this case.

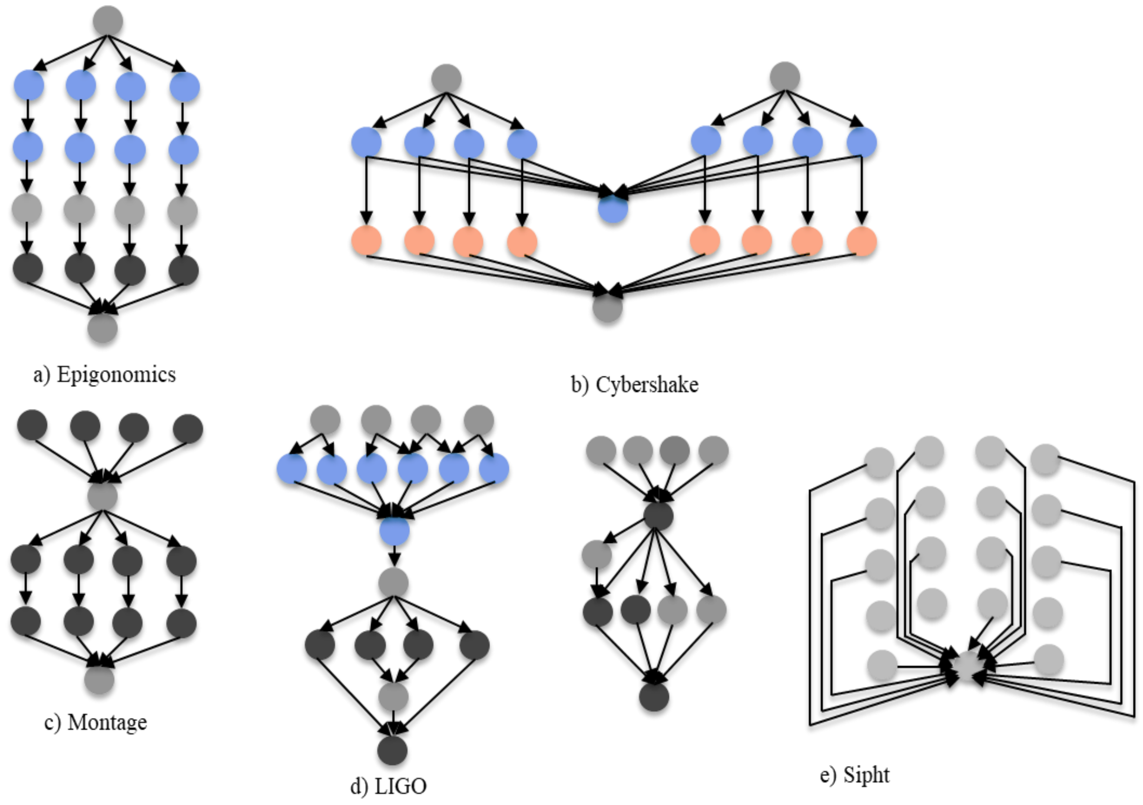


Fig. 3 Example of Scientific workflow [37,38,6]

Table 1. Requirement parameter

Parameter	Value
Simulator	iFogSim
Operating System	Windows 7 64-bit
MIPS	2000
No. Of Hosts	1 to 2
No. Of Fog Nodes	2 to 200
RAM	200 MB
Number of Workflows	10 - 12
Bandwidth	Up to 60 Mbps

Two hundred fog nodes make up the fog layer, separated across 10 clusters with 20 nodes each. Each machine needs 200 MB of Memory and a bandwidth of up to 60 Mbps. Ten to twelve processes with 100 to 1000 activities are taken into consideration to execute and analyze EERO.

3.3.5. The use of iFogSim Simulation

It is an open-source tool kit whose performance is considerably high. iFogSim is utilized in environments for edge, IoT, and fog computing. It is employed to model IoT networks and fog computing. Along with CloudSim, iFogSim operates. Three primary components make up iFogSim: logical components, which contain various application modules and application edges; physical components, which include physical fog nodes; and management components, which contain module mapping objects and a fog controller [39]. iFogSim is utilized in this work due to its simple user interface and minimal complexity. The primary CloudSim platform serves as the foundation for the iFogSim simulation toolkit. One of the most well-liked cloud computing simulators is CloudSim. IoT computers and numerous fog nodes can be used in iFogSim to simulate custom fog computation by extending the abstraction from basic CloudSim classes (e.g., sensors and actuators). To make it easier for users unfamiliar with CloudSim to recognize the fog computing architecture, service placement, and resource allocation policies, the groups are labelled in iFogSim. Every scenario in a fog computing environment is simulated by iFogSim using the fog nodes and sensors and distributed data flux paradigms. It facilitates the evaluation of network congestion, end-to-end latency, energy consumption, operating expenses, and quotas [32].

4. Results and Discussion

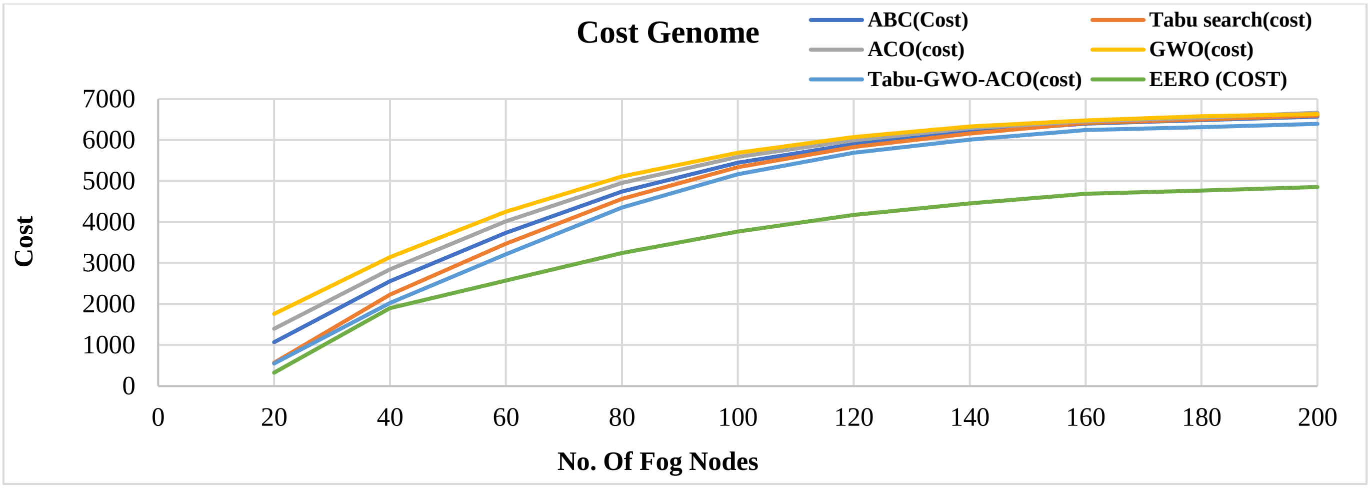
The outcomes of using the suggested algorithm will be presented in this section. Scientific workflow data sets are implemented utilizing simulation environments because executing them in a real-time context is difficult. To save time, money, and energy during execution in a fog environment, Eclipse is utilized to execute scientific workflow sets of data on the iFogSim simulator. This research takes a look at a few scientific workflows that implement the suggested method. The execution time, cost, and energy usage that were considered were all decreased when these procedures were implemented. Optimizing scientific workflow scheduling can make cloud computing much more efficient overall. As scheduling workflows are very much an NP-complete problem, meta-heuristic approaches are preferable for improving it [41]. Workflow benchmarks are essential for the development and evaluation of workflow management systems. Many scientific workflow data sets—such as Cybershake, LIGO, Sipht, and Genome are taken into account for experimental outcomes. The effectiveness of the EERO approach has been tested using the simulation data from iFogSim. Results from the proposed technique are compared to those from already-used techniques to confirm that EERO performs better than tabu search, ABC, GWO and ACO.

Test Study I: Analysis of Cost

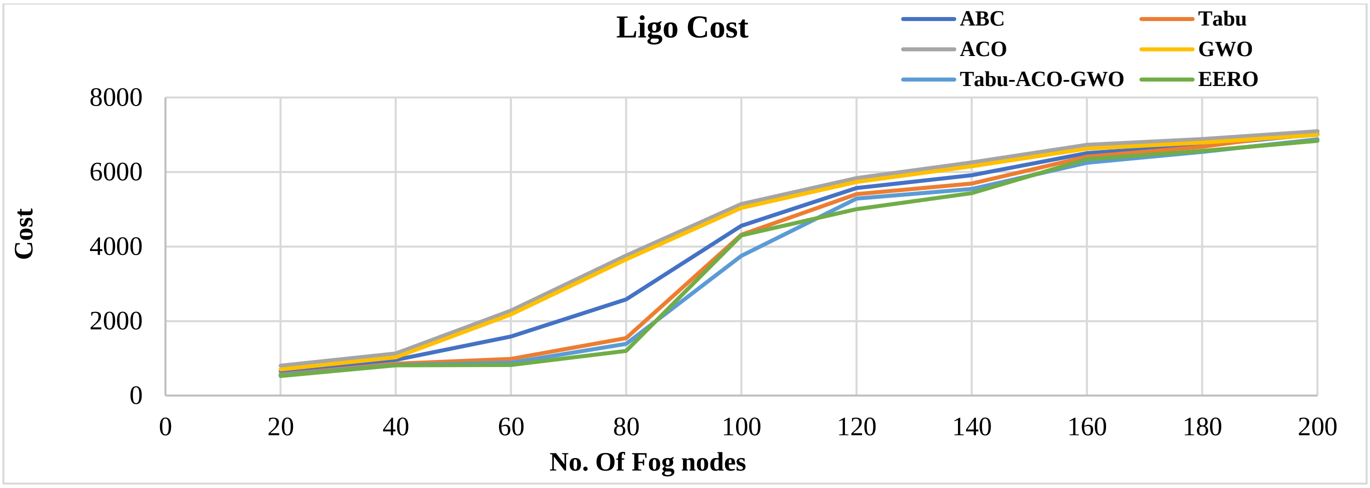
Fog nodes are given various types of workflow tasks, and their performance is evaluated. More fog nodes will be employed in the fog layer, increasing cost consumption. The execution workflows for Genome, Cybershake, Sipht, and LIGO were studied in this work, and various current methodologies were compared to the suggested one. Four subfigures of Figure 4 display various outcomes of the workflow execution. The results of the Genome scientific workflow execution are shown in Figure 4a. The graph displays the price on the y-axis and the number of fog nodes on the x-axis. Costs associated with implementation rise as the number of fog nodes rises. This article suggested an improved strategy based on PEFT ranking and the Bayesian approach.

The EERO approach that was suggested made an effort to lower the implementation cost for resource allocation. According to the figure, the proposed method is less expensive than other strategies like ACO, tabu search, ABC, and GWO. Other workflows, such as the Cybershake, Sipht, and LIGO tasks, have also been given to fog computing, and their findings have been archived. Figures 4a, b, c, and (d) exhibit these findings, respectively. Fog nodes are given full responsibility for all the outcomes produced by carrying out the various workflow activities, and their performance is evaluated. Fog nodes will cost more if used more frequently in the fog layer.

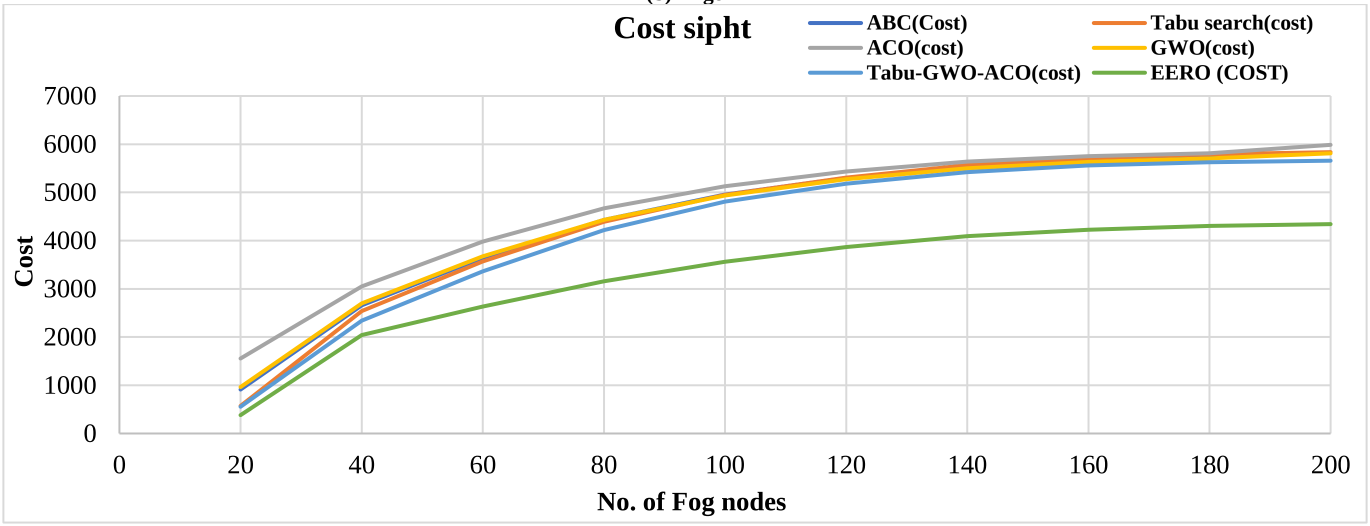
Genome, Cybershake, Sipht, and LIGO were four potential scientific workflows that were taken into consideration for implementation in this work, and existing methods and the suggested ones were compared. Using EERO instead of other existing approaches drastically reduces implementation costs for Genome and LIGO. For Sipht and Cybershake procedures, EERO reduces costs effectively compared to various current methods.



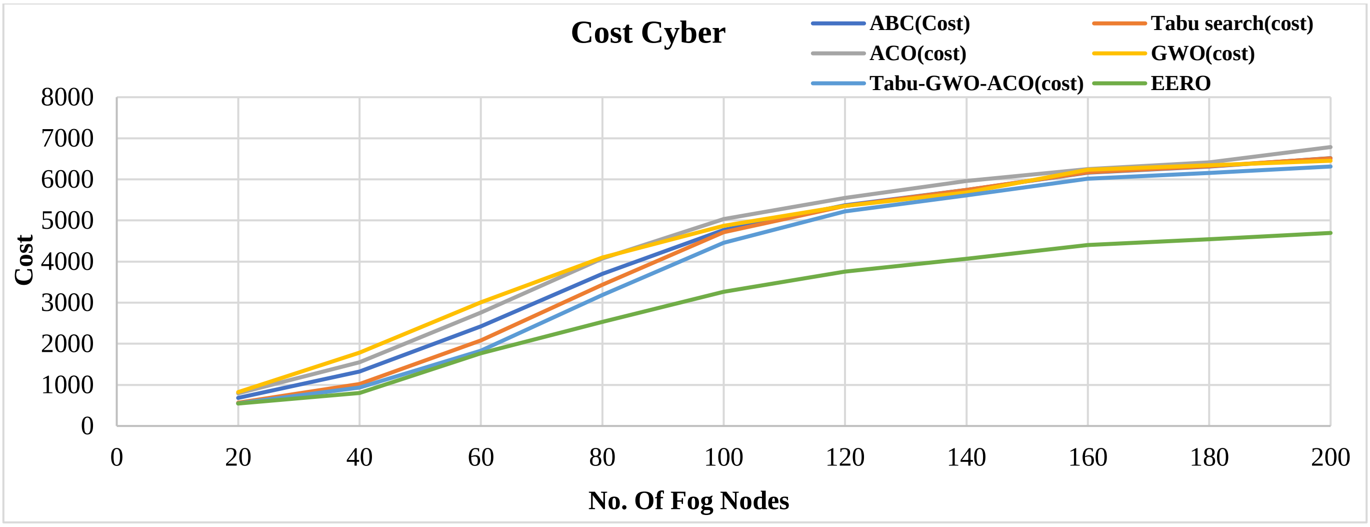
(a) Genome



(b) Ligo

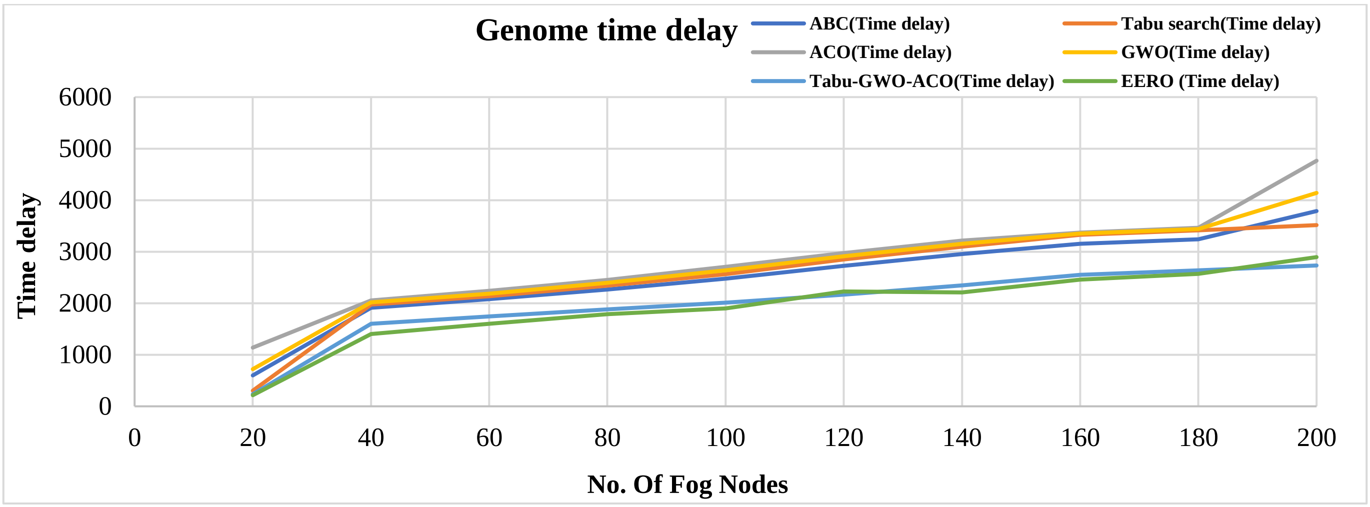


(c) Sipht

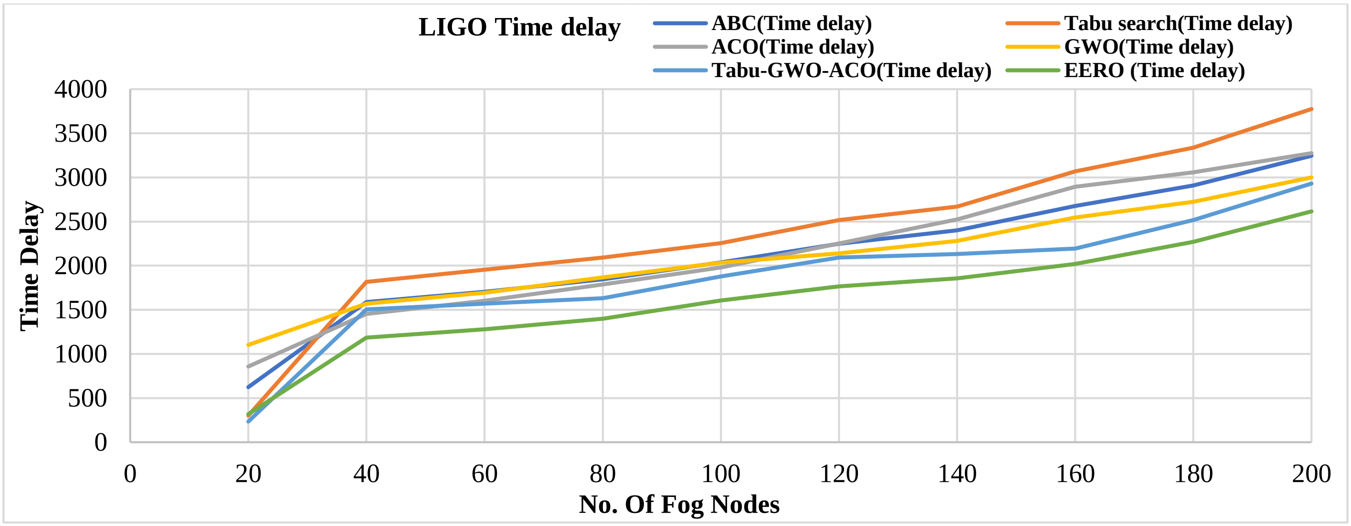


(d) CyberShake

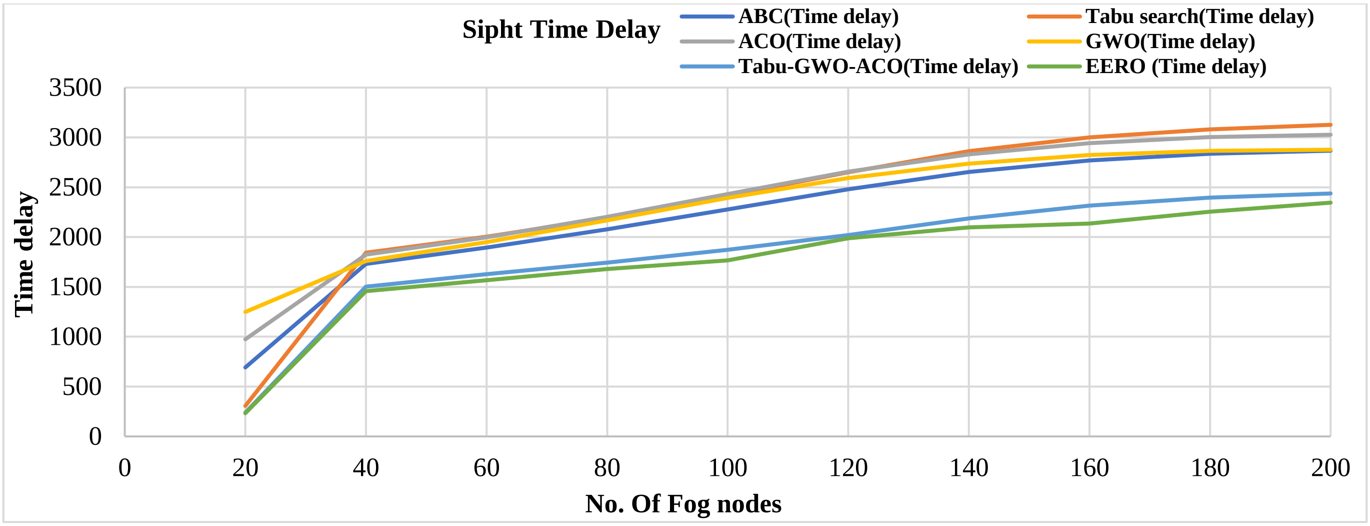
Fig. 4 Cost analysis of different workflows



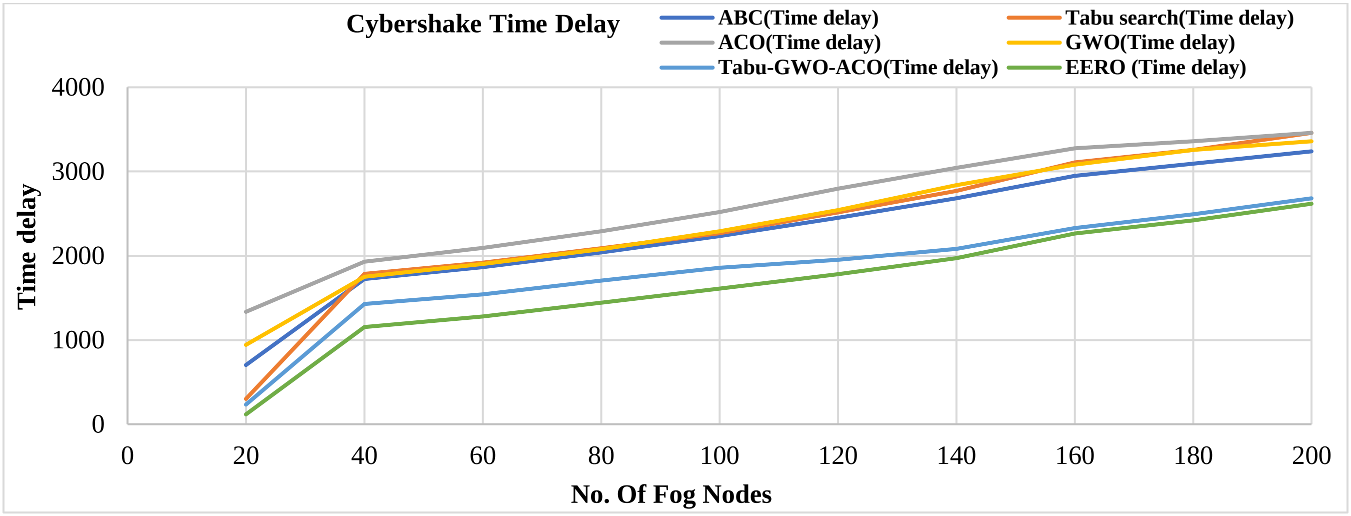
(a)Genome



(b) Ligo

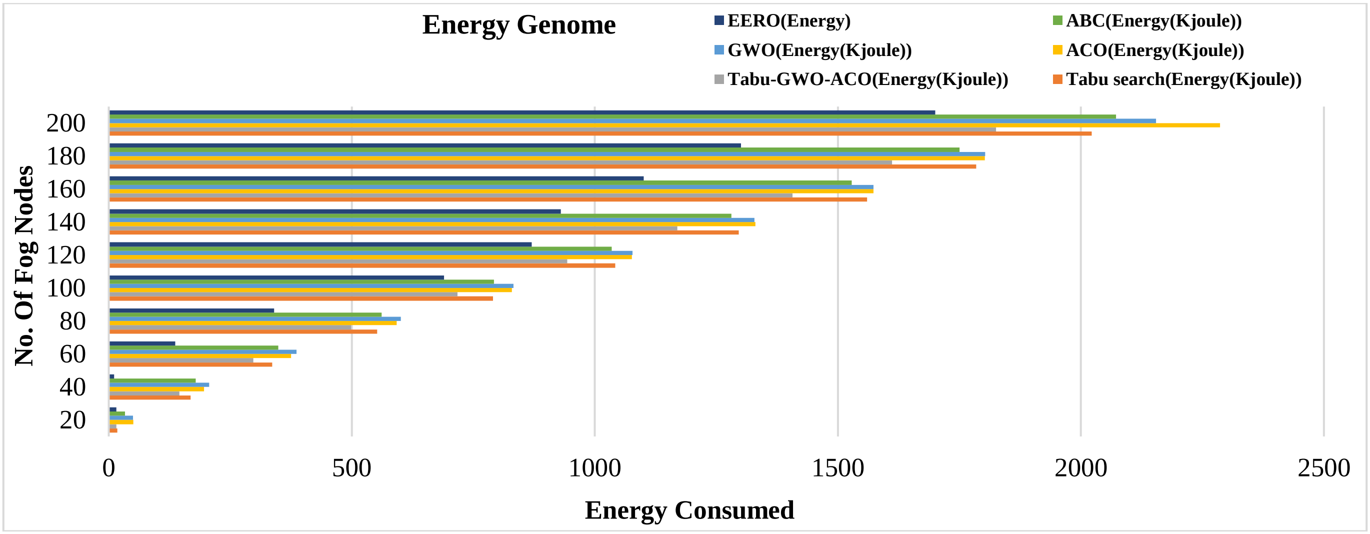


(C) Sipht

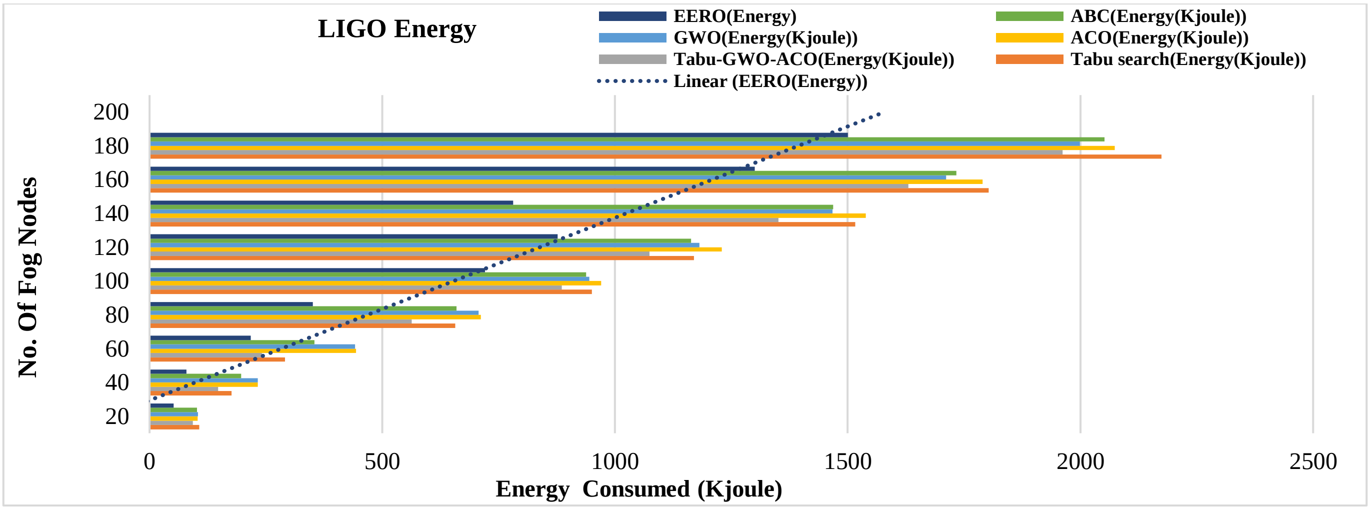


(d)CyberShake

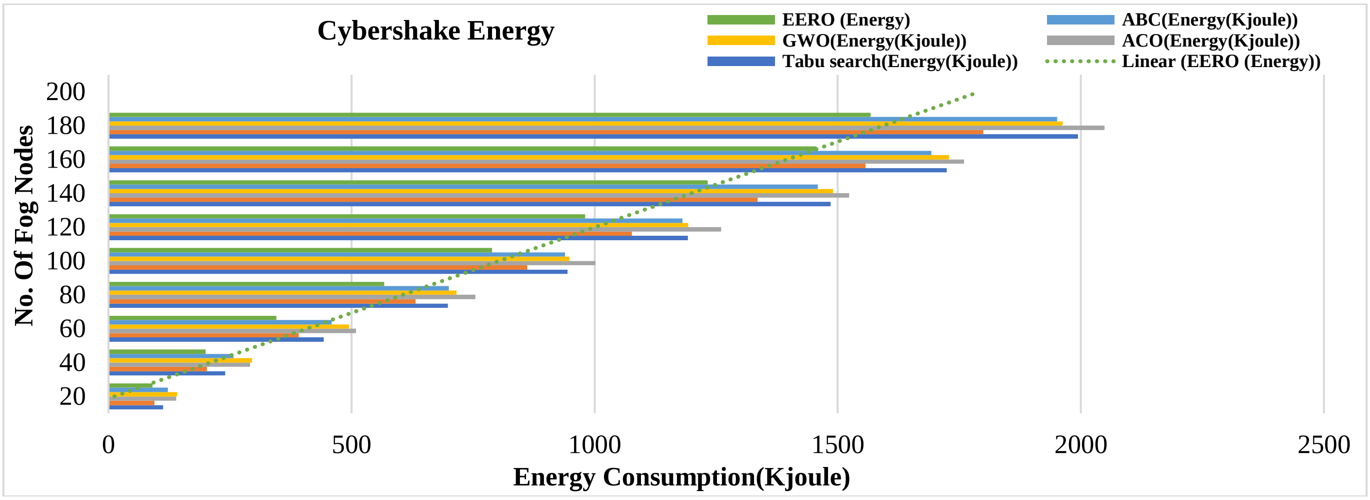
Fig. 5 Analysis of the execution times of various workflows



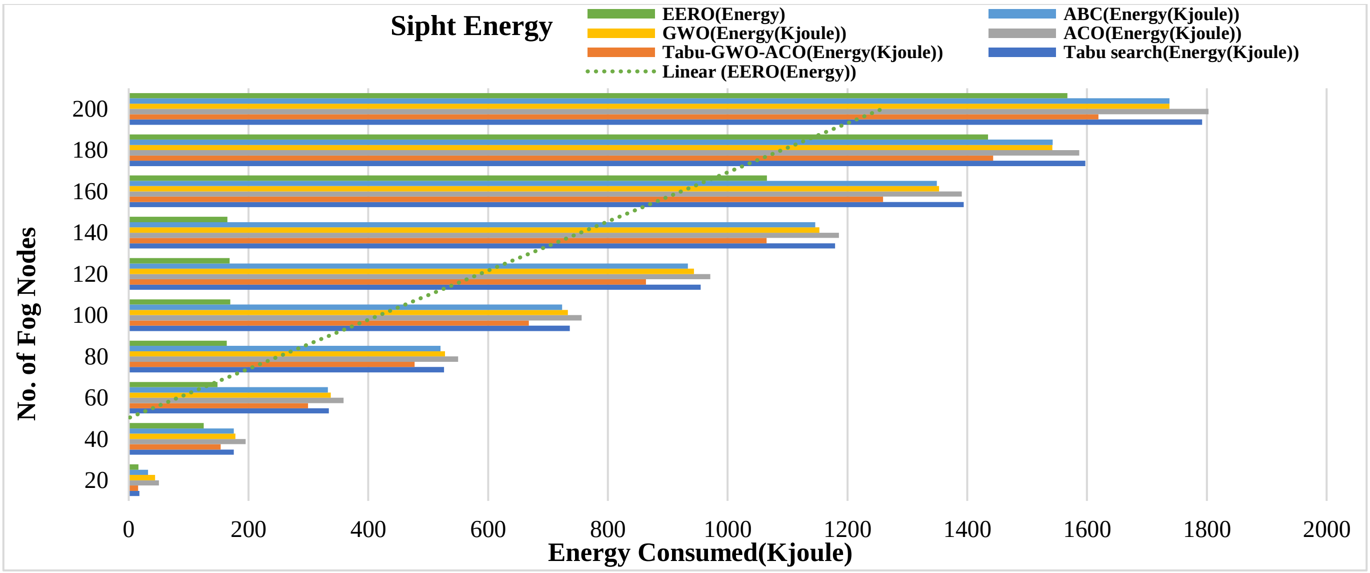
(a)Genome



(b) Ligo



(c) Cybershake



(d) Sipht

Fig. 6 Analysis of the energy use of various workflows

Test Study II Analysis of Execution Time

Large datasets are present in scientific operations like Genome, LIGO, Sipht, and Cybershake, where tasks are parsed to create fog nodes. To complete these significant duties, the fog layer requires more nodes. As the number of tasks rises, execution time grows. Task execution times in the fog layer have been examined after implementing the proposed approach. The analysis for the execution time is shown in Figure 5. of several process tasks. The time between task submission and job completion is used to compute task execution time. The length of time spent in line is also measured.

Figure 5 is divided into four sections, each representing the time taken for the scientific workflows represented by Genome, Sipht, LIGO and Cybershake. The y-axis measures execution time, and the x-axis displays the many fog nodes in the graphs. Graphs show that as the number of fog nodes increases, so does execution time. The suggested method attempted to shorten task execution times using the suggested EERO algorithm for scientific workflows. EERO reduces execution time in Genome and LIGO by 25% and 12%, respectively, compared to other current methods. The execution times of the Sipht and Cybershake workflows, however, are reduced by 18% and 20%, respectively, by EERO.

Test Study III Analysis of Energy Consumption

In this instance, the energy usage of multiple fog nodes in the fog layer is analyzed. More resources are needed when there are many jobs to complete. Energy usage increases together with the number of resources. The energy usage in fog nodes using the EERO technique is shown in Figure 6. The energy consumption in the fog layer is shown in Figures 6a, b, c, and d, where energy consumption is shown on the x-axis, and the number of fog nodes is shown on the y-axis. Since there are more tasks, there is a greater need for nodes, which increases energy usage. By balancing the load in the fog layer, the proposed strategy, called EERO, attempts to lower energy usage in fog nodes. The numbers demonstrate that EERO performs better than any other technique with which it is compared. For instance, EERO aims to cut the energy usage in fog nodes by 22.69% and 25%, respectively, in Genome and LIGO. On the other hand, energy use decreased by 25% and 24.56%, respectively, in Sipht and Cybershake.

5. Conclusion

An energy-efficient resource optimization technique has been presented for scientific workflow applications in the fog computing environment. First, we apply Pareto distribution to distribute tasks within budget and deadline efficiently. We employ a Bayesian method with a maximum likelihood procedure for processing the fog node tasks to find the Pareto front.

This article also proposes a resource management framework for fog computing. This study takes into account scientific workflow applications to evaluate the effectiveness of the suggested strategy. The outcomes of the suggested technique are compared to those of other current approaches to outperforming them. The primary goals of the suggested strategy are to reduce energy consumption and improve resource use. Security, resource provisioning, and energy usage are only unexplored issues that need to be studied thoroughly.These are a few current issues and potential directions:

– Fog node security and privacy have become significant issues in recent years. So, it is possible to think about this field for future research.
– There is a need for implementing resource optimization in a real situation happening in real-time.

These difficulties may inspire future scholars to investigate and develop the fog computing field. This study will be expanded to examine future problems related to the fog environment. The investigation will be expanded to load balancing with energy-conscious problems in the fog computing setting.

Funding Statement

Our work is conducted without any financial or non financial interests that could be perceived as potentially influencing the objectivity and integrity of the research.

References

[1] W.Z. Khan et al., “Industrial Internet of Things: Recent Advances, Enabling Technologies and Open Challenges,” *Computers & Electrical Engineering*, vol. 81, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[2] Mamta Agiwal, Abhishek Roy, and Navrati Saxena, “Next Generation 5G Wireless Networks: A Comprehensive Survey,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617-1655, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[3] Luiz André Barroso, and Urs Hölzle, “The Case for Energy-Proportional Computing,” *Computer*, vol. 40, no. 12, pp. 33-37, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[4] Mohammad Shojafar, and Mehdi Sookhak, “Internet of Everything, Networks, Applications, and Computing Systems (IoENACS),” *International Journal of Computers and Applications*, vol. 42, no. 3, pp. 213-215, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[5] Mandeep Kaur, and Rajni Aron, “FOCALB: Fog Computing Architecture of Load Balancing for Scientific Workflow Applications,” *Journal of Grid Computing*, vol. 19, no. 4, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Consortium for School Networking Initiative: Some Facts about Computer Energy Use, 2010. [Online] Available: http://www.cosn.org/Initiatives/GreenComputing/InterestingFacts/tabid/4639/Default.aspx

[7] Hendrik F. Hamann, Vanessa López, and Andrew Stepanchuk, "Thermal Zones for More Efficient Data Center Energy Management," *12th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, Las Vegas, NV, USA, pp. 1-6, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[8] Gerald Cabusao et al., "Data Center Energy Conservation Utilizing a Heat Pipe Based Ice Storage System," *IEEE CPMT Symposium Japan*, Tokyo, Japan, pp. 1-4, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[9] Zheng Chang et al., "Dynamic Resource Allocation and Computation Offloading for IoT Fog Computing System," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3348-3357, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[10] Xiaoge Huang et al., "Energy-Efficient Resource Allocation in Fog Computing Networks with the Candidate Mechanism," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8502-8512, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] Bushra Jamil et al., "A Job Scheduling Algorithm for Delay and Performance Optimization in Fog Computing," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 7, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[12] Arman Shehabi et al., "Data Center Growth in the United States: Decoupling the Demand for Services from Electricity Use," *Environmental Research Letters*, vol. 13, no. 12, pp. 1-12, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[13] Ruilong Deng et al., "Optimal Workload Allocation in Fog-Cloud Computing Towards Balanced Delay and Power Consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171-1181, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[14] Xuan-Qui Pham, and Eui-Nam Huh, "Towards Task Scheduling in a Cloud-fog Computing System," *18th Asia-Pacific Network Operations and Management Symposium*, Kanazawa, Japan, pp. 1-4, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[15] Artem M. Chirkin et al., "Execution Time Estimation for Workflow Scheduling," *Future Generation Computer Systems*, vol. 75, pp. 376-387, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[16] Vincenzo De Maio, and Dragi Kimovski, "Multi-Objective Scheduling of Extreme Data Scientific Workflows in Fog," *Future Generation Computer Systems*, vol. 106, pp. 171-184, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[17] Andras Markus, and Attila Kertesz, "A Survey and Taxonomy of Simulation Environments Modelling Fog Computing," *Simulation Modelling Practice and Theory*, vol. 101, pp. 1-18, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[18] Shaymaa Elsherbiny et al., "An Extended Intelligent Water Drops Algorithm for Workflow Scheduling in Cloud Computing Environment," *Egyptian Informatics Journal*, vol. 19, no. 1, pp. 33-55, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[19] Jian Qin, Brian Dobreski, and Brian Dobreski, "Metadata and Reproducibility: A Case Study of Gravitational Wave Research Data Management," *International Journal of Digital Curation*, vol. 11, no. 1, pp. 218-231, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[20] H. Topcuoglu, S. Hariri, and Min-You Wu, "Performance-Effective and Low-complexity Task Scheduling for Heterogeneous Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260-274, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[21] Hamid Arabnejad, and Jorge G. Barbosa "List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 682-694, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[22] Sayantani Basu et al., "An Intelligent/Cognitive Model of Task Scheduling for IoT Applications in Cloud Computing Environment," *Future Generation Computer Systems*, vol. 88, pp. 254-261, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[23] Lindong Liu et al., "A Task Scheduling Algorithm Based on Classification Mining in Fog Computing Environment," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–11, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[24] Deze Zeng et al., "Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702– 3712, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[25] Zhuo Tang et al., "An Energy-Efficient Task Scheduling Algorithm in DVFS-enabled Cloud Environment," *Journal of Grid Computing*, vol. 14, pp. 55-74, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[26] Kyong Hoon Kim, Rajkumar Buyya, and Jong Kim, "Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-Enabled Clusters," *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid*, Rio de Janeiro, Brazil, pp. 541-548, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[27] Philip Maechling et al., "SCEC CyberShake Workflows—Automating Probabilistic Seismic Hazard Analysis Calculations," *Workflows for e-Science*, pp. 143-163, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[28] Ying Xie et al., "A Novel Directional and Non-local-Convergent Particle Swarm Optimization Based Workflow Scheduling in Cloud–edge Environment," *Future Generation Computer Systems*, vol. 97, pp. 361-378, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[29] Scott Callaghan et al., "Reducing Time-To-Solution Using Distributed High-Throughput Mega-Workflows-Experiences from SCEC Cybershake," *IEEE Fourth International Conference on eScience*, Indianapolis, IN, USA, pp. 151–158, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[30] Ewa Deelman et al., "Managing Large-Scale Workflow Execution from Resource Provisioning to Provenance Tracking: The CyberShake Example," *Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*, Amsterdam, Netherlands, pp. 14-14, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[31] Heng Li, Jue Ruan, and Richard Durbin, "MAQ: Mapping and Assembly with Qualities," *Version*, vol. 6, no. 3, 2008. [Google Scholar]

[32] Sahar Saeedi et al., “Improved Many-Objective Particle Swarm Optimization Algorithm for Scientific Workflow Scheduling in Cloud Computing,” *Computers & Industrial Engineering*, vol. 147, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[33] Zhongjin Li et al., “A Security and Cost Aware Scheduling Algorithm for Heterogeneous Tasks of Scientific Workflow in Clouds,” *Future Generation Computer Systems*, vol. 65, pp. 140-152, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[34] Xiumin Zhou et al., “Minimizing Cost and Makespan for Workflow Scheduling in Cloud Using Fuzzy Dominance Sort Based HEFT,” *Future Generation Computer Systems*, vol. 93, pp. 278-289, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[35] Jonathan Livny et al., “Correction: High-Throughput, Kingdom-Wide Prediction and Annotation of Bacterial Non-Coding RNAs,” *PLoS ONE*, vol. 3, no. 11, pp. 1-12, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[36] Gideon Juve et al., “Characterizing and Profiling Scientific Workflows,” *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682-692, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[37] Maria A. Rodriguez, and Rajkumar Buyya, “Budget-Driven Scheduling of Scientific Workflows in IaaS Clouds with Fine-Grained Billing Periods,” *ACM Transactions on Autonomous and Adaptive Systems*, vol. 12, no. 2, pp. 1–22, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[38] Redowan Mahmud, and Rajkumar Buyya, “Modeling and Simulation of Fog and Edge Computing Environments Using iFogSim Toolkit,” *Fog and Edge Computing*, pp. 433–465, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[39] Harshit Gupta et al., “iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in the Internet of Things, Edge and Fog Computing Environments,” *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017. [CrossRef] [Google Scholar] [Publisher Link]