Article

Improved Performance and Cost Algorithm for Scheduling IoT Tasks in Fog–Cloud Environment Using Gray Wolf Optimization Algorithm

Naseem Adnan Alsamarai [1,2,*] and Osman Nuri Uçan [1]

1 School of Engineering and Natural Sciences, Electrical and Electronics Engineering, Altınbaş University, Istanbul 34218, Turkey; osman.ucan@altinbas.edu.tr
2 College of Alimam Aladham, Baghdad 10071, Iraq
* Correspondence: naseemadnan@imamaladham.edu.iq

Abstract: Today, the IoT has become a vital part of our lives because it has entered into the precise details of human life, like smart homes, healthcare, eldercare, vehicles, augmented reality, and industrial robotics. Cloud computing and fog computing give us services to process IoT tasks, and we are seeing a growth in the number of IoT devices every day. This massive increase needs huge amounts of resources to process it, and these vast resources need a lot of power to work because the fog and cloud are based on the term pay-per-use. We make to improve the performance and cost (PC) algorithm to give priority to the high-profit cost and to reduce energy consumption and Makespan; in this paper, we propose the performance and cost–gray wolf optimization (PC-GWO) algorithm, which is the combination of the PCA and GWO algorithms. The results of the trial reveal that the PC-GWO algorithm reduces the average overall energy usage by 12.17%, 11.57%, and 7.19%, and reduces the Makespan by 16.72%, 16.38%, and 14.107%, with the best average resource utilization enhanced by 13.2%, 12.05%, and 10.9% compared with the gray wolf optimization (GWO) algorithm, performance and cost algorithm (PCA), and Particle Swarm Optimization (PSO) algorithm.

Keywords: fog–cloud computing; task scheduling; energy consumption; Makespan; resource utilization; cost scheduling

check for updates

Citation: Alsamarai, N.A.; Uçan, O.N. Improved Performance and Cost Algorithm for Scheduling IoT Tasks in Fog–Cloud Environment Using Gray Wolf Optimization Algorithm. *Appl. Sci.* **2024**, *14*, 1670. https://doi.org/10.3390/app14041670

Academic Editors: Agostino Forestiero and Mustafa M. Matalgah

Received: 27 October 2023
Revised: 21 January 2024
Accepted: 29 January 2024
Published: 19 February 2024

1. Introduction

Cloud computing is a development in technology that emphasizes how we create apps, design computer systems, and use already-existing services to produce software. It is based on the idea of "dynamic provisioning", which applies to computing power, storage, networking, and infrastructure for information technology (IT) in general, as well as services. The National Institute of Standards and Technology (NIST) defined the cloud as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [1]. This cloud model is composed of five essential characteristics, which are on-demand self-service; three service models, which are Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS); and four deployment models, which are the private cloud, community cloud, public cloud, and hybrid cloud. Cloud services are provided to the Internet of Things devices (IoT); the IoT is a network of physical items (called "things") that are constructed with various sensors, software, and other technologies in order to connect with other devices and systems over the internet and share data with them. These gadgets range from simple household items to high-tech tools used in industry. Experts think that by 2025, there will be 75 billion connected IoT devices. Today, there are more than 15 billion IoT devices that are connected [2,3]. Energy consumption is a significant concern because a large number of devices need to be online. All these devices require a lot of data storage, computing power, and other resources [4] because of the time it takes for data to travel between the cloud and an IoT device. The applications that need real-time processing, like healthcare, vehicles, augmented reality, elder care, and industrial robots, are not easy to use with the cloud's storage and processing capabilities to obtain real-time services. Cisco proposed the idea of fog computing as a solution to this problem; it is essentially an expansion of cloud computing [5].

Fog computing was created to be a small cloud between the central cloud and IoT devices. The main advantage of this is that it reduces the distance between the cloud and IoT devices for real-time services. All of the client computers that are part of the fog network make up the devices layer of the architecture's top level. Between the endpoints and the cloud lies a layer of networking equipment known as the fog layer [6]. The cloud layer is depicted at the bottom, and the third layer is the cloud layer, which serves as the storage and processing hub for IoT devices. The proximity of the fog layer to the end devices means that edge devices can process Internet of Things (IoT) tasks much more quickly than the cloud [7]. Reduced latency, enhanced resource utilization, and reduced energy use are some crucial methods used to process them by building a scheduling algorithm to maximize system performance; however, sophisticated scheduling algorithms are required because the fog layer typically has fewer resources than the cloud. Task scheduling in cloud computing is an NP-hard problem. Therefore, we can maintain and build new scheduling algorithms to enhance the system's performance. Time and resource sharing, priority scheduling, and queuing approaches are just a few of the scheduling policies that can play a role in fog computing. In order to minimize power consumption and maximize response times for real-time applications, efficient scheduling is essential. In conclusion, the Internet of Things (IoT) is rapidly expanding, and the massive amounts of data it generates necessitate a lot of space, a lot of processing power, and a lot of other resources. The cloud can facilitate the management of such data streams, but it is not designed for use in real-time processing environments. Fog computing is a potential choice since it reduces query latency for IoT devices and makes the most efficient use of available resources. An attractive alternative for managing the growing number of IoT devices is fog computing due to the importance of efficient scheduling algorithms for optimizing system performance and resource utilization. Scheduling encompasses various policies, as outlined in ref. [8]:

- Increasing the workload's efficiency.
- Improving throughput.
- Decreased latency.
- Cost-cutting measures.
- Protecting the system from harm.
- Obtaining load balance.

The swarm intelligence paradigm is utilized by numerous artificial and biological systems [9]. The combination of the simple processes performed by many small, independent entities, driven by local information, allows for the emergence of sophisticated, intelligent behavior. The foraging behavior of ants, for instance, was studied. Another example is how birds are able to quickly adjust their motions to the changing characteristics of their environment through flocking behavior. A set of basic principles that individual birds obey can be defined to achieve this behavior. The authors in [10] proposed the gray wolf optimizer algorithm, inspired by the behavior of gray wolves to obtain prey. In this paper, we propose the PC-GWO algorithm, which is a hybrid algorithm, to enhance scheduling in a fog–cloud environment.

In this paper, we offer a unique scheduling method to assure the predictable execution of IoT activities and answer concerns about latency reduction, resource efficiency improvement, and decreased energy consumption. This method considers both task cost and power energy consumption constraints. This paper is structured as follows: Section 2 analyzes related work, and presents the system model, Our proposed algorithm is described and analyzed in Section 3, Section 4 explains the assessment, simulation model, and results, and Section 5 discusses the conclusion and future works.

2. Related Work

Efficient scheduling is critical to improving a system's performance. As a result, substantial research on fog–cloud computing job scheduling has been performed to address the challenge of distributing work to a collection of computers. Various approaches for task scheduling in fog–cloud computing have been developed, and a plethora of algorithms have been offered in the past to meet the issues connected to this environment. The following are some of these algorithms.

In ref. [11], in a cloud–fog scenario, the author describes a new technique for scheduling IoT device requests. This method involves incorporating Slap Swarm Algorithm (SSA) operators into an existing optimization methodology known as artificial ecosystem-based optimization (AEO). This move is intended to strengthen AEO's ability to address IoT-related concerns. The author also introduces ATS-FOA, a revolutionary collection of techniques that integrate algorithms such as approximate nearest neighbor (ANN) and fruit fly optimization (FOA) to reduce processing time, latency, memory utilization, and cost.

In ref. [12], the authors propose a new technique for scheduling tasks, which enhances firework algorithms. They suggest introducing a detection system for the explosion radius of fireworks. This approach can be employed in a cloud computing and fog computing system to reduce task processing time and achieve more effective load balancing across fog devices.

In ref. [13] researchers propose exploring serverless computing and its advantages for event-driven cloud applications. The proposed machine learning model utilizes gray wolf optimization (GWO) and reinforcement learning (RIL) to parallelize tasks and optimize task allocation in the serverless framework. The results of the simulation experiments show that the GWO-RIL approach reduces runtimes, adapts to varying load conditions, and achieves energy savings while maintaining similar latency. The findings suggest the potential for further development and research in distributed deep learning, task allocation for cloud-based and edge-based applications, and task offloading.

In ref. [6], the authors propose a new heuristic algorithm depending on the deadline and task bandwidth using the ACO algorithm for IoT task scheduling in a fog–cloud environment, optimizing Makespan and task deadline satisfaction. It prioritizes the earliest deadlines, assigning tasks to resources for minimum completion time. The evaluation shows superiority over existing methods in achieving low Makespan and high deadline satisfaction.

In ref. [14], the authors propose a new approach for the effective scheduling and activation management of fog computing services using the multi-objective gray wolf optimization (GWO) technique. This approach employs a multi-objective function to allocate resources and handle tasks, aiming to strike a balance between energy consumption and task execution time. The proposed method comprises two main stages: first, the GWO algorithm is utilized for scheduling, and then, container migration is employed for task offloading and resource allocation.

In ref. [15], researchers propose aims to address the problem of task scheduling in cloud computing environments, which is crucial for improving performance and scheduling tasks on virtual machines. The authors propose a modified gray wolf optimization (MGWO) algorithm, which utilizes a multi-objective fitness function that considers both Makespan and cost. By reducing both the cost and Makespan, the proposed technique aims to increase job scheduling efficiency.

In their work [16], the authors propose using Bumble Bee Mating Optimization (BBMO); the authors present a practical work scheduling approach for cloud computing. The strategy seeks to reduce Makespan when running tasks across several virtual machines (VMs). When compared to Honey Bee Mating Optimization (HBMO) and the Genetic Algorithm (GA), BBMO surpasses both. However, the additional complexity of BBMO necessitates more calculation time. Future studies will concentrate on optimizing the complexity of BBMO and investigating various consistencies in estimated task completion times (ETCs).

In ref. [17], the authors focus on load balancing in fog computing for healthcare applications. The authors present a probability-based strategy for determining job offloading to fog nodes with the least amount of latency. The fog master makes use of ant colony optimization (ACO) to assign tasks to appropriate nodes. Balancing load between fog nodes enhances the quality of service and reduces response time. The proposed work provides significant benefits for time-sensitive healthcare applications in fog computing.

In their publication [18], the researchers propose and the writers detail a technique for arranging tasks with multiple goals in fog computing, which utilizes an upgraded ant colony algorithm. This approach is customized to the features of fog nodes and accounts for expenses related to computing resources, power consumption, and network usage.

In this paper [19], PSO-GWO is a novel hybrid metaheuristic algorithm suggested for scheduling dependent tasks on virtual machines in cloud computing. To optimize the overall execution cost, the PSO-GWO technique combines Particle Swarm Optimization with gray wolf optimization. PSO-GWO beats the regular PSO and GWO algorithms in terms of average total execution cost and time, according to experimental data. When evaluated on scientific processes, the algorithm obtains significant percentage reductions in execution time and overall execution cost when compared to PSO and GWO, suggesting higher performance.

In this paper [20], the author introduces the Dynamic and Fault-Tolerant Scheduling Algorithm (DRFTSA). Leveraging task priorities based on unit electricity cost and a deep reinforcement learning model (DQN) within CloudSim, DRFTSA dynamically optimizes scheduling. Real-time Google Cloud Jobs data, categorized into regular and large datasets, are used for evaluation.

In this paper [21], the above-proposed method addresses the critical challenge of cloud task scheduling; this proposal introduces the Sine Cosine-based Elephant Herding Optimization (SCEHO) algorithm coupled with Improved Particle Swarm Optimization (IPSO), which enhances resource selection for tasks by incorporating load balancing and resource allocation parameters.

Table 1 provides a concise overview of the methodologies and critical parameters addressed by each task scheduling algorithm in the mentioned references. We can say that these works did not consider the huge amount of data that workflow tasks create or how quickly the fog–cloud computing world changes, which generates a huge amount of energy. Therefore, we propose a new algorithm that considers cost and energy in addition to Makespan to control the energy consumption and give the tasks priority depending on the term (pay as you go). Pay-as-you-go (PAYG) cloud computing is a way to pay for cloud computing that charges you based on how much you use it. It is kind of like how electricity bills work; they only use the resources they need [22].

Table 1. Examining current methods for scheduling tasks.

Reference	Methodology	Key Parameters
[11]	Incorporates SSA into AEO	IoT concerns, processing time, latency, memory, cost
[12]	Enhanced firework algorithms	Explosion radius detection, cloud–fog system, load balancing
[13]	GWO and RIL for serverless computing	Task parallelization, task allocation, runtime, energy
[6]	ACO algorithm for fog–cloud IoT scheduling	Makespan, task deadline satisfaction
[14]	Multi-objective GWO for fog computing	Energy consumption, task execution time, container migration
[15]	Modified GWO for cloud task scheduling	Makespan, cost, job scheduling efficiency
[16]	Bumble Bee Mating Optimization for cloud	Makespan, VMs, comparison with HBMO and GA
[17]	ACO for load balancing in fog computing	Latency, quality of service, response time
[18]	Upgraded ant colony algorithm for fog	Computing resource costs, power consumption, network usage
[19]	PSO-GWO hybrid for cloud task scheduling	Execution cost, time, performance
[20]	Dynamic and Fault-Tolerant Scheduling Algorithm	Task priorities, unit electricity cost, deep reinforcement learning
[21]	SCEHO and IPSO for cloud task scheduling	Load balancing, resource allocation

2.1. System Model

This portion provides an overview of the architecture of the IoT, fog computing, and cloud computing, and outlines the characteristics of the tasks that this system must handle.

Architecture

The structure of the IoT–fog–cloud system is depicted in Figure 1, beginning with the foundational layer of IoT devices. This layer comprises mobile devices, unmanned aerial vehicles (UAVs), medical devices, industrial equipment, smart homes, and other devices [23,24]. Data are generated as tasks and transferred to the fog or cloud computing layers. Fog computing acts as an intermediary between the IoT and the cloud, sharing similar features but with restricted computing capabilities. The cloud, on the other hand, has vast resources for storage and processing [25]. "Cloud computing is a technological advancement that focuses on the way we design computing systems, develop applications, and leverage existing services for building software. It is based on the concept of dynamic provisioning, which is applied not only to services but also to compute capability, storage, networking, and information technology (IT) infrastructure in general" [26]. To ensure the timely completion of critical tasks, such tasks are sent to the fog level, whereas tasks with lower costs can be directed to the cloud to optimize resource allocation.

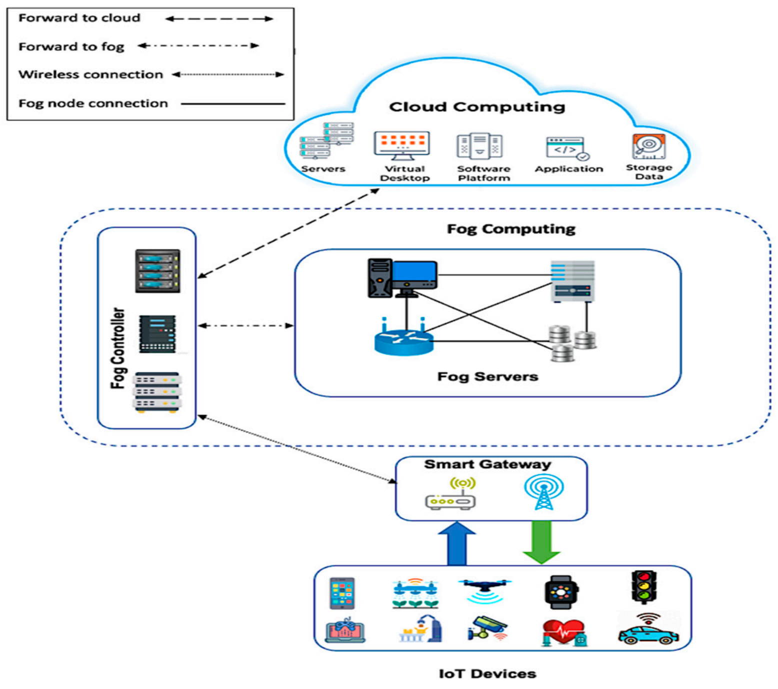

Figure 1. System architecture.

A fog broker [13,27,28] is the entity responsible for handing the assignment of a particular job to either the fog or cloud, and it is deployed within the fog layer. The fog controller is made up of three essential parts:

(1) Task director (TD): this component receives the task and analyzes its characteristics.
(2) Resource monitoring (RM): it gathers information about the available resources.
(3) Task scheduler (TS): this component receives the task and resource information from the TD and RM, respectively, and then determines the appropriate resource for task execution.

Once the task scheduler assigns tasks to the suitable resources, the processing takes place, and upon completion, the results are sent back to the original sender.

2.2. Problem Definitions

As all tasks in the fog–cloud share identical attributes, we proceeded to outline these task characteristics using mathematical equations.

2.2.1. Execution Time

The duration required for task T_i to execute by resource R_j can be determined using Equation (1):

$$EX_{(i,j)} = \frac{t_{s(i,j)}}{r_{p(i,j)}} \tag{1}$$

where $t_{s(i,j)}$ represents the task (t_i) size that needs to be executed, and $r_{p(i,j)}$ represents the power computation of resource *j*.

$R = Set\ of\ resources,\ where \mid R \mid = m$
$T = Set\ of\ tasks,\ where \mid T \mid = n$
$j = Index\ of\ resources,\ j \in R$
$i = Index\ of\ tasks, i \in T$

2.2.2. Completion Time

The completion time corresponds to the duration required for resource R_j to complete the processing of task T_i. Equation (2) computes the completion time:

$$CT_{ij} = EX_{(i,j)} + r_j \tag{2}$$

In this case, r_j denotes the ready time of resource R_j. In some situations, the resources in the fog or cloud may not be instantly available, requiring the job to wait until the resource is ready for processing. This waiting period is known as the readiness time, and it is combined with the execution time to establish the completion time.

2.2.3. Makespan

The difference in time between the commencement and completion of a series of work or tasks, denoted as "t_i", can be determined through the utilization of Equation (3).

$$makespan = max(CT_i)_{t_i \in MetaTask} \tag{3}$$

By and large, scheduling with the minimum Makespan is preferable.

2.3. Calculate the Priority Depending on the Task Cost

In step one, we calculate the cost of each activity for each available resource using Equation (4):

$$Cs\left(t_{iR_j}\right) = NI(t_i) \times\ CI\left(t_{iR_j}\right) + TB(t_i) \times CPBW\left(t_{iR_j}\right) \tag{4}$$

where

$NI(t_i)$ is the number of job instructions (t_i),
$CI\left(t_{iR_j}\right)$ is the expense of t_i on resource per instruction R_j,
$TB(t_i)$ is the data for a task (t_i),
and $CPBW\left(t_{iR_j}\right)$ is the cost of performing the job per bandwidth T_i on resource R_j.

The profit is then determined for each job using Equation (5) for the resource with the most significant cost.

$$Profit = CsU(T_i) – CO\ \left(T_{iR_{max}}\right) \tag{5}$$

where

$CsU(T_i)$ is the cost paid by the user to run task T_i,
$CO\left(T_{iR_{max}}\right)$ is the actual cost of running task T_i on the resource R_{max} (R_{max} is the resource that has the highest cost).

Subsequently, the tasks are organized and allocated to their respective queues. In our study, we introduced three levels of queues (Q1, Q2, and Q3) with equal priority ranges.

This means that the overall priority range of the latest tasks is evenly distributed among these queues. Consequently, utilizing Equation (6), we can compute the rounded-up queue range (Q_r) of priorities for each queue since we have four queues in total.

$$Q_r = Pr_{max}(T_i)/3 \tag{6}$$

where $Pr_{max}(T_i)$ can be paraphrased in the text as follows:

In PCA, "the highest profit received from completing task T_i is divided by three to account for the presence of a three queue. The algorithm consists of three queues, each with a different priority region: the High queue, which has the most significant priority zone; the Middle queue, which has a medium priority region; and the Low queue, which has the lowest priority region. As previously said, this enables us to view [7]:

IF ($Pr(T_i) \leq Q_r$), THEN the task T_i is added to the Q3.
IF ($Q_r < Pr) \leq 2Q_r$) THEN, the task T_i is added to the Q2.
IF ($2Q_r < Pr(T_i) \leq 3Q_r$) THEN the task T_i is added to the Q1.

2.4. GWO Algorithm

The gray wolf optimizer (GWO) is a metaheuristic algorithm inspired by the social structure and hunting behaviors of gray wolves. In the wild, gray wolves have a hierarchical structure, with an alpha leader and lower-ranking beta and delta wolves. To effectively seek prey, these wolves collaborate as a team. Likewise, the GWO method is intended to tackle optimization problems by emulating the cooperative and communicative behaviors of wolves. The various solutions to the problem are modeled as a pack of wolves, and their placements in the solution space represent their fitness. The alpha wolf is currently the most excellent solution, although the beta and delta wolves are also viable options [10].

The description of the mathematical representation for the GWO (gray wolf optimization) algorithm is outlined below.

Encircling Prey: This phase involves the gray wolves encircling their prey. Mathematically, this behavior is represented by the following equations, Equations (4) and (5):

$$\vec{D} = \vec{C}.\vec{X}_p(t) - \vec{X}_w(t) \tag{7}$$

$$\vec{X}_w(t+1) = \vec{X}_p(t) - \vec{A}.\vec{D} \tag{8}$$

where t stands for the current iteration, $\vec{A}$ and $\vec{C}$ are the coefficient vectors, the prey's position vector is indicated by X_p, and $\vec{X}_w$ represents a gray wolf's position vector. In addition, $\vec{A}$ and $\vec{C}$ are calculated using the following formulas: Equations (9) and (10).

$$\vec{A} = 2\vec{a}.\vec{r_1} - \vec{a} \tag{9}$$

$$\vec{C} = 2.\vec{r_2} \tag{10}$$

where r_1 and r_2 depict a set of random vectors in [0, 1], and $\vec{a}$ is linearly decreased from 2 to 0.

Hunting: During this stage, the alpha wolf, serving as the leader, and the beta and delta wolves, acting as consultants, possess ample knowledge about the prey's whereabouts. As a result, the other wolves must adjust their positions based on the location of the most skilled agent, represented by mathematical equations Equations (11)–(13):

$$\vec{D}_\alpha = \left|\vec{C}_1 \,.\, \vec{X}_a - \vec{X}\right| \quad \vec{D}_\beta = \left|\vec{C}_1 .\vec{X}_\beta - \vec{X}\right| \quad \vec{D}_\delta = \left|\vec{C}_1 .\vec{X}_\delta - \vec{X}\right| \tag{11}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1.\vec{D}_\alpha \quad \vec{X}_2 = \vec{X}_\beta - \vec{A}_2.\vec{D}_\beta \quad \vec{X}_3 = \vec{X}_\delta - \vec{A}_3.\vec{D}_\delta \tag{12}$$

$$\vec{X_w}(t+1) = \frac{X_1 + X_2 + X_3}{3} \tag{13}$$

Attacking Prey: Exploitation and exploration, which involve pursuing and searching for prey, are essential aspects of wolves' hunting behavior. When wolves chase and attack prey, it showcases their proficiency in capturing prey, which can be considered as their exploitation ability. This exploitation ability allows wolves to reach global optima, maximizing their success in hunting. The value of parameter A plays a vital role in this process. If the magnitude of ∣A∣ < 1, the gray wolves are compelled to pursue the current prey aggressively. However, if the magnitude of ∣A∣ > 1, the gray wolves are compelled to abandon the current prey and explore other potential targets. The pseudocode of the GWO algorithm [10] is depicted in Algorithm 1.

Equations Power Consumption Models: The energy usage of a server can be categorized into two components: idle and dynamic. Idle consumption refers to the power used when the server is inactive and not performing any tasks, while dynamic consumption represents the power utilized during active operations and computations. We can calculate the idle power consumption and dynamic power consumption given by Equations (14) and (15), respectively [29,30].

Algorithm 1: Pseudocode of GWO Algorithm [10]

```
1.   "Initialize Population
2.   Initialize a, A, and C
3.   Calculate the fitness of each search agent.
4.   X_α = the best search agent
5.   X_β = the second-best search agent
6.   X_δ = the third-best search agent
7.   While (t < maximum number of iterations)
8.   For each search agent
9.   Update the position of the current search agent by.
10.  End for
11.  Update a, A, and C
12.  Calculate the fitness of the current search agent.
13.  Update Best Solution.
14.  Update X_α, X_β and X_δ
15.  t= t + 1
16.  End While
17.  Return X_α"
```

$$P_{C_{idle}} = \sum_{i=1}^{n} P_{r_i} \tag{14}$$

where P_{ri} represents the reduced power consumption of R_i.

$$P_{C_{dynamic}} = \sum_{i=1}^{n} P_i' \tag{15}$$

where P_i denotes the dynamic power consumption of the R_i having a utilization (load) of L_i.

Total power consumption: A multi-core processor's total power consumption is defined by Equation (16):

$$P_c = P_{C_{idle}} + P_{C_{dynamic}} \tag{16}$$

Fitness Function: In this paper, we consider two parameters: the first one is the minimum completion time, and the second one is the power consumption. The fitness function (FF) can be calculated using Equation (17).

$$FF = CT_{ij} + P_c \tag{17}$$

3. Our Proposed Algorithm

The energy consumption of these data centers, currently amounting to 2% of global power usage, or 416.2 terawatt hours, is estimated to increase dramatically. By the year 2022, projections suggest it will make up 7% of the world's energy consumption, indicating a significant surge in demand [31]. Therefore, we propose a new algorithm that considers the cost and energy, in addition to the Makespan, of task execution by enhancing the PCA algorithm; our algorithm first applies the PCA algorithm, as shown in the pseudocode [8] in Algorithm 2. When the algorithm arrives at step four, the GWO algorithm is called, as shown in Algorithm 3, which represents the pseudocode of PC-GWO algorithm.

Algorithm 2: Pseudocode of PCA Algorithm [8]

```
"1-FOR all available tasks DO
Calculate the priority of each task.
END FOR

2-Sort the tasks according to the priorities in the scheduler's queues.

3-FOR all tasks Ti in meta-task DO
    FOR all resources Rj DO
Calculate the completion time:
(CT)_ij = $(EC)_{ij}+r_j$
END FOR
END FOR

4-Find task Tk which has the highest Priority and assign this task Tk to the resource which has the minimum
completion time.

5-Remove task Tk from Meta-tasks set and update rj for the selected Rj and Update CTij for all j.

6-IF the waiting time of any task in the lower queues has exceeded the threshold THEN
Move this/these tasks to the next upper queue.
END IF.

7-IF there is a new task has arrived THEN Calculate its priority and sort it in the end of appropriate queue and
repeat the above steps
END IF"
```

Algorithm 3: Pseudocode of PC-GWO Algorithm

```
Step 1-FOR all tasks in meta task DO
Calculate the priority of each task.
END FOR

Step 2-insert the tasks according to the priorities in the scheduler's queues.

Step 3-FOR all tasks Ti in meta-task DO
    FOR all resources Rj DO
Calculate the completion time:
(CT)ij = (EC)ij+rj
END FOR
END FOR

Step 4-Call the GWO with the fitness function using equation (17)

Step 5-Strip task $T_k$ from Meta-tasks set and update rj for the selected Rj and update CTij for all j.

Step 6-IF the waiting time for any task in the lower queues has gone over the threshold, THEN Shift this/these
tasks to the next upper queue.
END IF.

Step 7-IF there is a new task has arrived THEN Calculate its priority and place it at the end of the right queue
and repeat the steps above.
END IF
```

Where:

Step 1: Calculate the priority of each task in the meta-task using Equation (5).
Step 2: Sort the tasks in the queue according to their priority using Equation (6).
Step 3: Calculate the completion time for all tasks in the meta-task for each available resource.
Step 4: Call the GWO algorithm to choose the best solutions to process the task using the fitness function by using Equation (17).
Step 5: Remove the task that was processed from the meta-task, update the ready time of resource j, and update the completion time of all resources.
Step 6: In this step, we consider the waiting time of the task in the queue to determine if it will arrive at the deadline or not by migrating the task, which is estimated to arrive at the next level.
Step 7: In this step, if a new task is coming, the algorithm repeats all the steps above.

4. Performance Evaluation and Result

This section focuses on evaluating the performance of the proposed algorithms in comparison to other algorithms. The evaluation is based on three key metrics: Makespan, cost, and energy consumption. The simulation implementing and evaluating the suggested approach necessitates a set of programming tools to execute various scenarios, along with a comprehensive comparison of the proposed solutions. In this chapter, we first explore the implementation of the proposed solution (PC-GWO). Using the Java programming language, we created a rudimentary version of the algorithm. This simulation was made on a computer equipped with an Apple M1 chip with 8 GB of memory and the operating system macOS Ventura 13.2. To conduct the experimental testing, three different scenarios were created with different values of priority to check the system performance in different cases to be sure if it was potentially influenced by fortuitous initial conditions or not.

Scenario 1: There are numerous tasks of high priority, accompanied by a smaller number of tasks with medium and low priority.

Scenario 2: There are numerous tasks of medium priority, along with a few tasks of high and low priority.

Scenario 3: There are numerous tasks of low priority, along with a few tasks of high and medium priority.

Tables 2 and 3 show the cloud and fog characteristics, respectively, and the simulation and algorithm of GWO parameters are shown in Table 4.

Table 2. Cloud Parameters.

Parameter	Value
Cloud nodes number	5
Processing rate of the cloud nodes	[20,000, 40,000] MIPS
Cloud nodes bandwidth	[1500, 4000] MBPS
Latency from fog to cloud (ms)	100
Power consumption in idle mode	(65–75) % of the active dynamic mode
Power consumption in dynamic mode	(100–200) Watts

Table 3. Fog Parameters.

Parameter	Value
Fog nodes number	(10, 20, 30)
Processing rate of the fog nodes	[5000, 20,000] MIPS
Fog nodes bandwidth	[5000, 10,000] MBPS
Latency from the device to fog layers (ms)	20
Deadlines of tasks (ms)	[50–150]
Power consumption in idle mode	(65–75) % of the active dynamic mode
Power consumption in dynamic mode	(75–150) Watts

Table 4. Simulation and GWO algorithm.

Parameter	**Value**
The number of tasks	[50, 100, 150, 200]
Number of wolves	30
Number of iterations	25
C	[0, 2]
r_1, r_2	[0, 1]
D	Any Value

For the purpose of demonstrating the efficacy of our proposed algorithms, we compared them with the benchmarks that follow.

(1) **Performance and Cost Algorithm (PCA):** Performance and cost scheduling algorithm for cloud services. The (PCA) algorithm is introduced, which prioritizes tasks based on profits to optimize resource utilization and minimize Makespan. The PCA considers both completion time and cost priority to provide cost-effective services with enhanced performance for cloud users.

(2) **The Gray Wolf Optimizer:** The gray wolf optimizer (GWO) algorithm is a meta-heuristic algorithm that was inspired by gray wolves' social structure and hunting activity. Gray wolves have a hierarchical structure in nature, with an alpha leader and lower-ranking beta and delta wolves.

(3) **Particle Swarm Optimization (PSO):** PSO is a clever approach that mimics the behavior of a flock of birds. Particles represent the birds in the context of the issue being addressed in this technique, and a multidimensional velocity governs their movement. A particle's position is impacted at each iteration by both the best position identified thus far and the best position among all particles in the whole problem space. A fitness function evaluates each particle's fitness value, showing its proximity to the desired aim in the search space and, hence, its relevance. Furthermore, each particle has a velocity that influences its path. Each particle successfully explores the issue space by continually pursuing the most promising particles at any given time [32].

The results of our experiments are presented on this page. First, we look into the impact of the algorithms we propose on performance. Following that, we compare these algorithms to existing methods by altering the number of tasks, fog nodes, and cloud nodes. Figure 2 depicts the Makespan for the four approaches under the conditions mentioned above. We note that our algorithm minimizes the Makespan compared with the GWO, PCA, and PSO algorithms, 16.72%, 16.38%, and 14.107%, respectively. Our proposed technique beats the other three algorithms (GWO, PCA, and PSO) in all circumstances. Furthermore, Figure 3 depicts the average resource usage for the four algorithms (PC-GWO, GWO, PCA, and PSO) in scenarios A, B, and C, each with 10, 20, and 30 for fog and five nodes of cloud for each scenario: 94%, 72%, 78%, and 85%; 90%, 70%, 75%, and 83%; and 92%, 74%, 76%, 84%, respectively, clearly demonstrating that our suggested approach outperforms GWO, PCA, and PSO in terms of resource utilization. Finally, Figure 4 displays the average energy in kilojoules used for the four algorithms (PC-GWO, GWO, PCA, and PSO) in scenarios A, B, and C, each with 10, 20, and 30 for fog and five nodes of cloud for each scenario: 22.75, 24.75, 26.25, and 23.55; 32.5, 35.89, 34.7, and 33.88; and 39.6, 47.25, 46.32, and 44.77, respectively. This shows that our proposed method is better than GWO, PCA, and PSO in this area. We created three scenarios in our study to measure the performance of our suggested algorithm. Scenario one included a more significant number of high-priority tasks and a lesser number of low- and medium-priority tasks.

Scenario two included a significant number of tasks of medium priority, as well as a few tasks of high and low priority. Scenario three had several low-priority tasks, as well as a few high- and medium-priority tasks.

We employed three node number phases, the first with ten nodes, the second with twenty, and the third with thirty. Each stage contained five cloud nodes with a 100-millisecond delay in the cloud and a 20-millisecond delay in the fog. The number of tasks ranged from 50 to 200 in all circumstances.

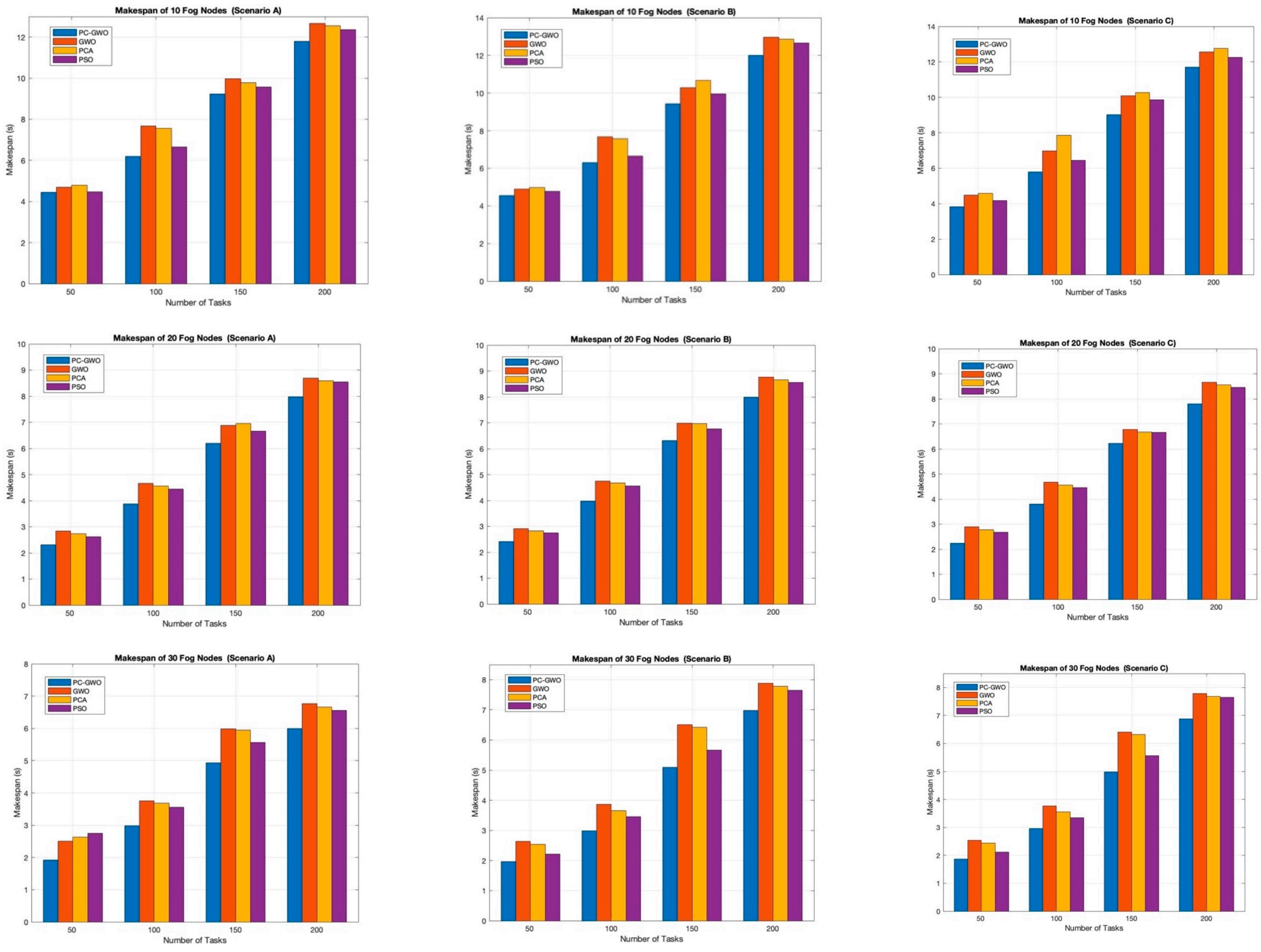

Figure 2. Display of the tasks' Makespan for each scenario (A, B, and C).

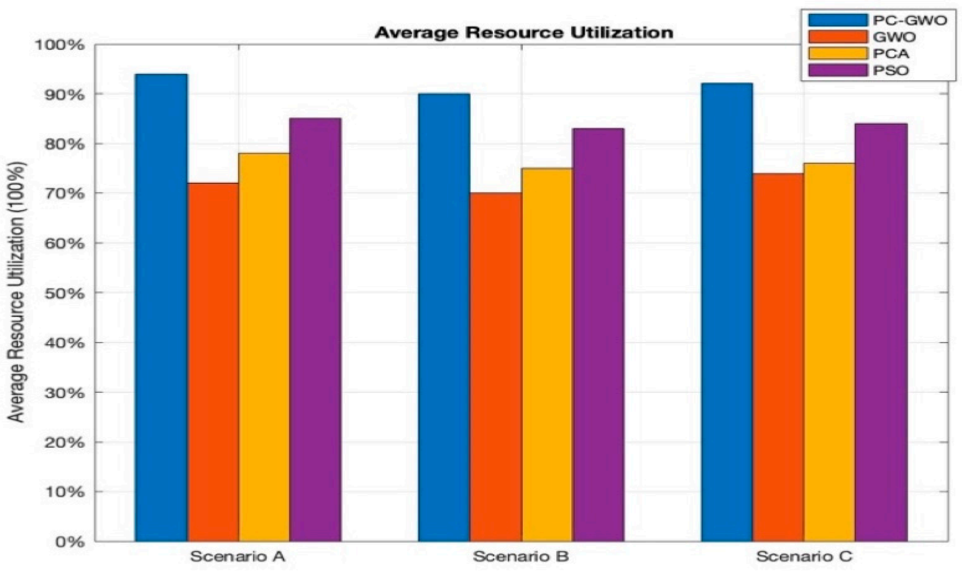

Figure 3. Average Resource Utilization.

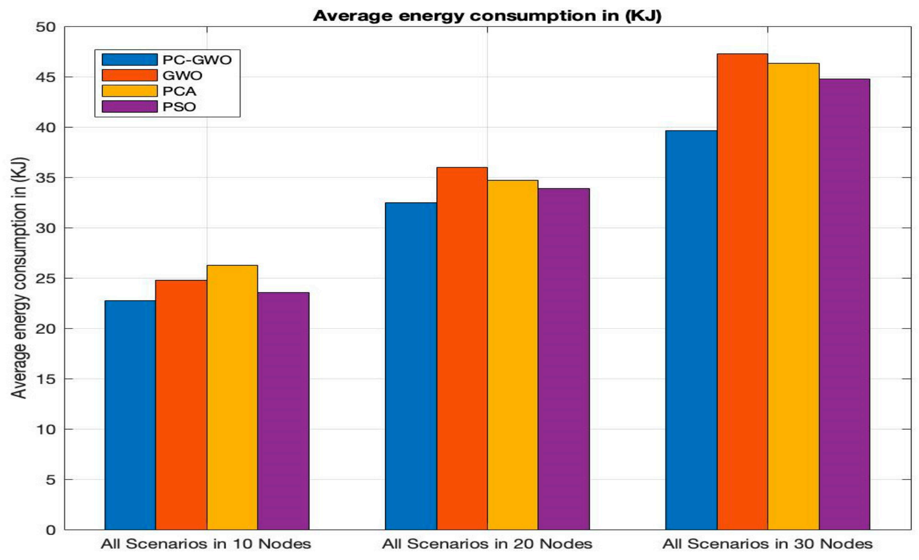

Figure 4. Average Energy Consumption (KJ).

The results of the experiment demonstrated that our algorithm performed exceptionally well in contrast to the other algorithms that were investigated. Our findings are supported by Figures 2–4, which show that, in three scenarios, our proposed algorithm is superior to the alternatives in terms of Makespan, average resource utilization, and energy power usage. This benefit results from the evaluation that our algorithm performs on job priority, in the cost field, the minimal completion time, the energy power consumption, and the efficient distribution of work across the fog and cloud environments, using the GWO algorithm to determine the optimal solution. In addition to this, it determines the quickest route between Internet of Things devices and the fog–cloud environment, which further contributes to its efficiency.

5. Conclusions

Within the scope of this investigation, we investigated the IoT job scheduling method within a fog–cloud setting. The primary objective of this research was to increase the system's overall Makespan by enhancing the performance and cost algorithm (PCA) with gray wolf optimization (GWO) methodologies. This algorithm considers the cost-to-profit ratio and allocates the task that would yield the most significant profit to the resource that would accomplish it in the least amount of time while also consuming the least amount of energy. The new algorithm was put through a battery of tests to determine its efficacy, and the results of the trial reveal that the PC-GWO algorithm reduces the average overall energy usage by 12.17%, 11.57%, and 7.19%, and reduces the Makespan by 16.72%, 16.38%, and 14.107%, with the best average resource utilization by 13.2%, 12.05% and 10.9% compared with the gray wolf optimization (GWO) algorithm, performance and cost algorithm (PCA), and Particle Swarm Optimization (PSO) algorithm. In the future, there will be a need to address plenty of issues, such as the temperatures of the system and the times of the deadlines.

Author Contributions: N.A.A.—writing the draft, simulating results; O.N.U.—conceptualization, validating results. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: All study data will be provided based on the request. The data is not publicly available as we use a prioritised scheduler to evaluate our metrics, which cannot be openly shared.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Mell, P.; Grance, T. *The NIST Definition of Cloud Computing*; National Institute of Standards & Technology: Gaithersburg, MA, USA, 2011.
2. Babun, L.; Denney, K.; Celik, Z.B.; McDaniel, P.; Uluagac, A.S. A survey on IoT platforms: Communication, security, and privacy perspectives. *Comput. Netw.* **2021**, *192*, 108040. [CrossRef]
3. What Is the Internet of Things (IoT)? Available online: https://www.oracle.com/internet-of-things/what-is-iot/ (accessed on 5 August 2023).
4. Basmadjian, R.; De Meer, H.; Lent, R.; Giuliani, G. Cloud computing and its interest in saving energy: The use case of a private cloud. *J. Cloud Comput. Adv. Syst. Appl.* **2012**, *1*, 1–25. [CrossRef]
5. Buyya, R.; Srirama, S.N. *Fog and Edge Computing: Principles and Paradigms*; John Wiley & Sons: Hoboken, NJ, USA, 2019.
6. Alsamarai, N.A.; Uçan, O.N.; Khalaf, O.F. Bandwidth-Deadline IoT Task Scheduling in Fog–Cloud Computing Environment Based on the Task Bandwidth. *Wirel. Pers. Commun.* **2023**. [CrossRef]
7. Azizi, S.; Shojafar, M.; Abawajy, J.; Buyya, R. Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach. *J. Netw. Comput. Appl.* **2022**, *201*, 103333. [CrossRef]
8. AL-Sammarraie, N.; Alrahmawy, M.; Rashad, M. A Scheduling Algorithm to Enhance the Performance and the Cost of Cloud Services. *Int. J. Intell. Comput. Inf. Sci.* **2015**, *15*, 1–14. [CrossRef]
9. Cicirelli, F.; Forestiero, A.; Giordano, A.; Mastroianni, C. Transparent and efficient parallelization of swarm algorithms. *ACM Trans. Auton. Adapt. Syst.* **2016**, *11*, 1–26. [CrossRef]
10. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
11. Elaziz, M.A.; Abualigah, L.; Attiya, I. Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments. *Future Gener. Comput. Syst.* **2021**, *124*, 142–154. [CrossRef]
12. Wang, S.; Zhao, T.; Pang, S. Task scheduling algorithm based on improved firework algorithm in fog computing. *IEEE Access* **2020**, *8*, 32385–32394. [CrossRef]
13. Yuvaraj, N.; Karthikeyan, T.; Praghash, K. An improved task allocation scheme in serverless computing using gray wolf Optimization (GWO) based reinforcement learning (RIL) approach. *Wirel. Pers. Commun.* **2021**, *117*, 2403–2421. [CrossRef]
14. Hashemi, S.M.; Sahafi, A.; Rahmani, A.M.; Bohlouli, M. Gwo-sa: Gray wolf optimization algorithm for service activation management in fog computing. *IEEE Access* **2022**, *10*, 107846–107863. [CrossRef]
15. Alzaqebah, A.; Al-Sayyed, R.; Masadeh, R. Task scheduling based on modified grey wolf optimizer in cloud computing environment. In Proceedings of the 2nd International Conference on New Trends in Computing Sciences (ICTCS), Amman, Jordan, 9–11 October 2019; pp. 1–6.
16. Alotaibi, M.T.; Almalag, M.S.; Werntz, K. Task Scheduling in Cloud Computing Environment Using Bumble Bee Mating Algorithm. In Proceedings of the IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT), Dubai, United Arab Emirates, 12–16 December 2020; pp. 1–6.
17. Alotaibi, B.K.; Broudi, U. Offload and Schedule Tasks in Health Environment using Ant Colony Optimization at Fog Master. In Proceedings of the International Wireless Communications and Mobile Computing (IWCMC), Dubrovnik, Croatia, 30 May–3 June 2022; pp. 469–474.
18. Gu, J.; Mo, J.; Li, B.; Zhang, Y.; Wang, W. A multi-objective fog computing task scheduling strategy based on ant colony algorithm. In Proceedings of the IEEE 4th International Conference on Information Systems and Computer Aided Education (ICISCAE), Dalian, China, 24–26 September 2021; pp. 12–16.
19. Arora, N.; Banyal, R.K. A particle grey wolf hybrid algorithm for workflow scheduling in cloud computing. *Wirel. Pers. Commun.* **2022**, *122*, 3313–3345. [CrossRef]
20. Krishna, M.S.R.; Mangalampalli, S. A Novel Fault-Tolerant Aware Task Scheduler Using Deep Reinforcement Learning in Cloud Computing. *Appl. Sci.* **2023**, *13*, 12015. [CrossRef]
21. Rajashekar, K.J.; Channakrishnaraju; Gowda, P.C.; Jayachandra, A.B. SCEHO-IPSO: A Nature-Inspired Meta Heuristic Optimization for Task-Scheduling Policy in Cloud Computing. *Appl. Sci.* **2023**, *13*, 10850. [CrossRef]
22. Huang, J.; Susilo, W.; Guo, F.; Wu, G.; Zhao, Z.; Huang, Q. An Anonymous Authentication System for Pay-As-You-Go Cloud Computing. *IEEE Trans. Dependable Secur. Comput.* **2020**, *19*, 1280–1291. [CrossRef]
23. Tripathy, B.K.; Anuradha, J. *Internet of Things (IoT): Technologies, Applications, Challenges and Solutions*; CRC Press: Boca Raton, FL, USA, 2017.
24. Siozios, K.; Anagnostos, D.; Soudris, D.; Kosmatopoulos, E. *IoT for Smart Grids*; Springer: Cham, Switzerland, 2019.
25. Chang, W.; Wu, J. *Fog/Edge Computing For Security, Privacy, and Applications*; Springer: Berlin/Heidelberg, Germany, 2021.
26. Buyya, R.; Vecchiola, C.; Selvi, S.T. *Mastering Cloud Computing: Foundations and Applications Programming*; Newnes: Lithgow, NSW, Australia, 2013.
27. Adhikari, M.; Mukherjee, M.; Srirama, S.N. DPTO: A deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing. *IEEE Internet Things J.* **2019**, *7*, 5773–5782. [CrossRef]
28. Omer, S.; Azizi, S.; Shojafar, M.; Tafazolli, R. A priority, power and traffic-aware virtual machine placement of IoT applications in cloud data centers. *J. Syst. Archit.* **2021**, *115*, 101996. [CrossRef]

29. Jia, M.; Chen, W.; Zhu, J.; Tan, H.; Huang, H. An Energy-aware Greedy Heuristic for Multi-objective Optimization in Fog-Cloud Computing System. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 11–14 October 2020; pp. 794–799.
30. Lin, W.; Wang, H.; Zhang, Y.; Qi, D.; Wang, J.Z.; Chang, V. A cloud server energy consumption measurement system for heterogeneous cloud environments. *Inf. Sci.* **2018**, *468*, 47–62. [CrossRef]
31. Khalil, M.I.K.; Shah, S.A.A.; Taj, A.; Shiraz, M.; Alamri, B.; Murawwat, S.; Hafeez, G. Renewable-aware geographical load balancing using option pricing for energy cost minimization in data centers. *Processes* **2022**, *10*, 1983. [CrossRef]
32. Baburao, D.; Pavankumar, T.; Prabhu, C.S.R. Load balancing in the fog nodes using particle swarm optimization-based enhanced dynamic resource allocation method. *Appl. Nanosci.* **2021**, *13*, 1045–1054. [CrossRef]