Al-Rafidain Journal of Engineering Sciences

Journal homepage https://rjes.iq/index.php/rjes

ISSN 3005-3153 (Online)

Optimized Task Offloading in Hybrid Cloud-Fog Computing for IoT Using Horse Herd Optimization Algorithm (HHOA)

Rawaa Nadhim Kadhim

Master of Computer Engineering, Azad University of Isfahan, Isfahan, Iran

ARTICLE INFO

Article history:
Received 02 September 2025
Revised 02 September 2025 ,
Accepted 20 October 2025,
Available online 29 October 2025

Keywords:
Task Offloading
Hybrid Cloud–Fog Computing
Internet of Things (IoT)
Horse Herd Optimization Algorithm (HHOA)
Resource Utilization.

ABSTRACT

One of the main obstacles to enhancing the functionality is task offloading for Internet of Things (IoT) applications in hybrid cloud and fog computing environments. This study looks into an effective task offloading strategy that uses the Horse Herd Optimization Algorithm (HHOA) to reduce execution time and resource costs while ensuring balanced resource use. Unlike most previous studies that depend on synthetic datasets, this work uses a real-world IoT fog and cloud dataset sourced from Kaggle, which makes the evaluation more realistic and relevant. The dataset includes 120 tasks spread across 13 computing nodes, featuring 10 fog nodes and 3 cloud nodes, each with different processing capacities and resource costs. Simulation results show that HHOA achieves a total execution time of 0.9731 seconds, an optimal scheduling cost of 2.155, and a makespan of 6.1296 seconds. The highest recorded execution time per task was 0.8757 seconds. The highest execution cost per task was 0.8712. This shows minimal variation and balanced load distribution. The algorithm smartly assigned time-sensitive tasks to cloud nodes for quicker processing. It allocated less urgent tasks to fog nodes to preserve their limited CPU, RAM, and power resources. These findings confirm that HHOA effectively delivers a cost-efficient, performance-optimized, and resource-conscious method of job offloading for cloud computing and hybrid IoT fog settings.

1. Introduction

Smart cities will emerge from congested ones thanks to the IoT, or Internet of Things. A smart city's objective is to enhance the standard of services provided to citizens by utilizing state-of-the-art IT technology. Numerous geographically dispersed nodes that comprise the Internet of Things collaborate to fulfill the smart city's objective. Every smart sensor on every node has the ability to recognize a physical aspect of the surroundings that is within its transmission range. It combines and evaluates the gathered data to produce decisions and coordinated actions. Smart homes, video surveillance, emergency response, traffic control, smart transportation, and health care are just a few of the many IoT application areas [1]. Because it may provide processing and storage capabilities that no IoT device would otherwise be able to access, analytics and storage are backed up by cloud computing. In this context, task offloading refers to the process of storing the workload (set of created data) produced by Internet of Things devices on the cloud [2-4]. Offloading tasks to the cloud layer is ineffective in this case and would result in network bandwidth overheads due to the multiple levels of redundancy that offer the ability to filter out a significant amount of the data. Additionally, because of the comparatively high network latency, shifting IoT jobs to the cloud slows down data processing reaction times. In order

Corresponding author E-mail address: rawaaalisawi256@gmail.com
https://doi.org/10.61268/fd21fn46

to bring processing power closer to Internet of Things devices, the fog computing idea was created. Numerous computing paradigms are employed to manage and store the massive volumes of data produced by an expanding number of devices. Cloud, edge, and fog computing are the three most popular and commonly utilized computing paradigms. Three computing tiers of cloud, edge, and fog computing systems are referred to by the words cloud, edge, and fog. The three layers of the cloud-edge-fog system are depicted in Figure 1. (a) Cloud Tier: The cloud computing paradigm, which is the most popular and widely used computing paradigm for over ten years because of its alluring features like scalability, rapid elasticity, resource pooling, cost savings, and ease of maintenance, is found in the cloud system [6], the top tier. This tier includes a number of clouds, including Google and Amazon. These focus mostly on business logic, analytical databases, industrial big data, data "warehousing," and other related topics. (b) Edge tier: An edge system is the middle layer in the edge computing paradigm [7]. When the European Telecommunication Standards Institute (ETSI) suggested that mobile network providers host cloud computing's capabilities, edge computing was born. This tier, which is home to several service providers like AT&T, Verizon, TMobile, Chunghwa Telecom, and others, is made up of base stations, central offices, micro-data offices, and local network assets. (c) Fog tier: a fog system is the lowest tier [8] or Internet of Things system, which includes, among other things, smart sensors, industrial actuators, Internet of Things devices, mobile users (including laptops, smartphones, and tablets), and smart cars that generate fog in their vehicles [9]. It consists of autonomous machinery, process-specific software, electronic vehicles, a collection of local computing devices, and industrial PCs that process data in real time.

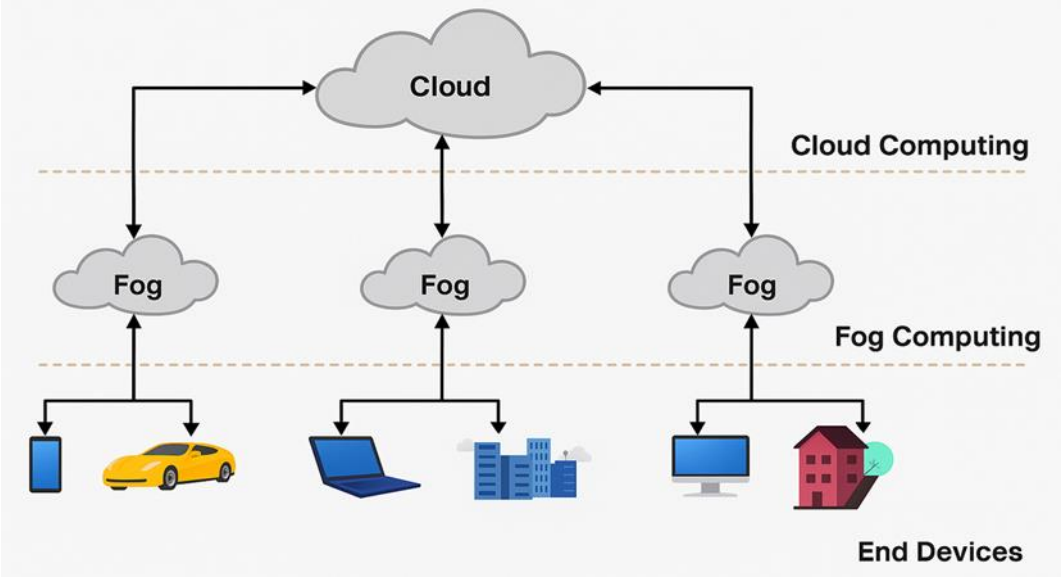

Figure 1.Cloud–fog–edge computing integration

By lowering network latency on the cloud, fog computing increases data analysis response time and minimizes high network congestion, making it a good choice for IoT task offloading use cases. Since the IoT devices are situated near the network's edge, where computational power, faster response times, and low latency are available, this is also known as edge computing. However, between fog computing and edge computing, there are some distinctions. Fog computing uses network equipment on the LAN to which the IoT devices are attached (e.g., routers, gateways) to provide computing, networking, and storage services for the devices. Edge computing offers computation and storage capabilities using comparatively tiny data centers near IoT devices, as opposed to the cloud, which uses

massive data centers located far from the devices connected by Wi-Fi access points [10]. For a number of sophisticated applications, such as autonomous driving, traffic control, and health care apps, fog computing offers quality of service (QoS). To distribute the tasks among the fog nodes and offload work while maintaining the necessary quality of service (QoS) of the jobs, including their response times, it is prudent to select the right fog nodes.

Task offloading refers to the shifting of a task or workload from a nearby device to a distant one, like a server or cloud resource, in order to enhance the effectiveness and performance of the local device. Task offloading could then also incur additional latencies and energy use. Edge servers frequently have low power, increasing latency impact. Identify the potential trade-offs involved before making a choice. High-density 5G networks may incur additional transmission delays themselves. Cooperative task offloading is one specific edge-cloud network paradigm that may improve distributed system performance. In this paradigm, tasks are segmented among network devices, such as edge devices and cloud servers. This can be a more efficient use of resources as opposed to a fully utilizing any single device. In an edge-cloud computing environment when you want to choose the best location to offload a task is difficult. This is because each location will have different computing capabilities of edge servers, different transmission delays of networks, and then end devices will have different requirements. There is a number of studies that focus on computation offloading in edge-cloud networks [11–15]. However, edge-cloud networks introduce a similar offloading problem, without knowledge of the intended computation taken by end devices and limited knowledge of wireless channels with respect to bandwidth and computing resources obstruct effective offloading strategies to consider.

The majority of IoT nodes have limited computing power and limited resources [16]. One intriguing technique that potentially boost application performance is the ability to transfer workloads from devices with limited resources to fog computing. Fog computing has two advantages: first, it uses less energy and reaction time; second, it minimizes the data-transmission latency when jobs are offloaded to fog servers rather than cloud servers. For a number of reasons, fog computing is a suitable solution for task offloading in Internet of Things applications: it facilitates time-sensitive applications at the network's edge; by offering services at the edge, it drastically lowers network workload; and it is scalable and diverse [17].

Despite the fact that task offloading in hybrid cloud and fog computing environments is advantageous, there are still many issues that need to be resolved, such as choosing between various offloading strategies in dynamic heterogeneous networks, the limited energy of IoT devices, and the variations in processing and storage capacities. The guarantee of low latency for time-sensitive applications, data security and privacy, and problems with scalability and mobility are some of the other difficulties. To overcome these obstacles and encourage dependable and effective IoT performance, clever solutions will be needed.

The aim of this research is to create a more efficient work offloading strategy for fog and hybrid cloud computing-based Internet of Things (IoT) applications. The algorithm (HHOA) will be used to accomplish the study's objectives, which include lowering costs and energy consumption, improving execution delay, and increasing the application's resource efficiency.

2. Related Work

Task offloading refers to executing computations or tasks using low-powered devices (such as IoT-type devices) and shifting them to the remote server [18], [19], in order to achieve the necessary quality of service (QoS). The situation in which all IoT devices send to the cloud creates a significant amount of traffic that leads to network congestion as well as latency. In order to successfully deal with these latencies, fog computing technology will shift computations as close to the IoT devices at the

network's edge, therefore reducing the response time by way of latencies. Response time is critical to real-time applications, including video streaming, smart transportation, and self-driving cars.

A lot of research has been done on offloading tasks to the edge of an IoT network. According to [20], the authors suggest a task-offloading framework to help IoT-fog-cloud apps reduce service latency. Based on a technique that can reduce service delays, the authors develop a system that takes into consideration the kinds of workloads produced by Internet of Things devices and the burdens placed on fog nodes. A fog node will accept the task if the service latency, as indicated by its current load, falls below a predefined threshold. If not, the task will be sent to the closest appropriate fog node. Out of all the neighbor nodes, the one with the lowest predicted propagation and service delays will be chosen. Lastly, the job will be offloaded to the cloud if it reaches an offloading upper bound. However, the delay reduction policy does not take into consideration any communication costs because the fog nodes were presumed to be physically connected to one another in this work.

A linear programming heuristic for the job allocation problem in medical cyber-physical systems was introduced by the authors of [21]. Linear programming uses a heuristic approach to find the best work distribution based on the performance of virtual machines used by virtual medical devices in fog computing environments. The heuristic technique takes account of the expenses of connectivity, computing, virtual machine deployment, and task distribution to maintain the essential level of quality of service (QoS) of real-time medical applications.

The fog nodes are thought of as cellular network base stations in the suggested strategy, which is based on a cellular network architecture. Just the location of virtual machines is displayed because there isn't any real task dumping or load balancing. As a result, a fog computing-based architecture cannot use the technique.

The authors of [22] looked into how to divide the workload across the cloud and fog nodes while reducing power usage and adhering to service delay restrictions. However, the research assumed that the fog nodes were collaborating with the cloud computing servers, furthermore, it only considered the fog nodes as separate points of contact with the cloud nodes.

An interface between the fog nodes and embedded devices for job management and offloading is suggested in [23]. Reducing Software Defined Embedded Systems' processing time is the primary goal. In particular, a mixed-integer nonlinear programming joint optimization model for task scheduling and storage placement is the suggested approach.

In [24], a basic task offloading architecture based on job classification and a basic cost function is advised for Internet of Things applications that are information-centric. The cost of communication was not included in this paper. An approach that includes into account where work offloading should be placed on the fog nodes is shown in [25] utilizing a cost function that takes power, computation, and communication expenses into account.

An orchestration for task offloading services was presented in [26] for mobile edge computing on the Internet of Things. Software defined networking (SDN) is used in this service orchestration technique to lessen the load on network transmission. Additionally, based on an optimization function of energy consumption for communication, energy consumption for computing, and delay models of the separate jobs, the authors offer a heuristic approach for differentiated cloud-edge offloading options. In order to meet network requirements, the SDN framework in [27] decreases service reaction time while simultaneously lowering data redundancy in the cloud-edge layer.

In [28], [29] a cloud-fog architecture is investigated, which is also coordinated in a smart grid (SG) context for resource management, and where requesting scheduling from smart devices (e.g., smart meters) that connect to fog node's virtual machines were energized through a variety of hybrid techniques, such as ACO, particle swarm optimization (PSO), round robin, and throttled, as well as an ACO algorithm and artificial bee colony hybrid that performed better than all other hybrid approaches. However, limitations do exist in this research investigation, especially in relation to a smart grid scenario's cloud-fog architecture and how there is no exact clarity in the description of the proposed algorithms as well as the load balancing method. Related study on the application of an ACO algorithm for IoT task scheduling on SG fog nodes can also be found in [30].

The goal of the anticipatory algorithm we presented was to shorten the IoT tasks' response times. It is crucial to remember that our suggested ACO method depends on IoT task profiling. They suggested a graph-based algorithm in [31] to aim for work load balancing across the fog nodes. However, given that the strain on the fog nodes can be temporary, we thought this was overly theoretical. In [32], they highlighted striking a balance between memory and computation time and suggested a bee life method to plan the IoT job processing on the fog nodes.

In order to balance computation time and operating costs, they suggested using a genetic algorithm in [33] to plan the IoT task processing on the fog nodes. Using a PSO, the same kind of work was given in [34], although none of these studies took communication costs into account. An associated issue is the best location of fog devices in a given broad area to improve fog node power consumption and leverage the Internet of Things context, while also ensuring that the QoS requirements of the Internet of Things-based application are satisfied. For example, [35] use evolutionary algorithms to achieve a workable trade-off for workload offloading between the fog node, power consumption, and QoS. PSO with a multi-objective function was used in [36] to solve the edge server placement problem by lowering overall energy consumption while preserving a manageable access delay. To find the optimal offloading option with the lowest latency, a hybrid genetic-simulated annealing latency-latency minimization offloading choice technique is devised in [37] for IoT-fog computing. Furthermore, a genetic algorithm has been developed for offloading choices in IoT-fog computing [38]. However, none of these techniques take advantage of the fog nodes' need for load balancing.

In this study, we tackle the difficulties of offloading IoT tasks using a hybrid cloud–fog computing approach to lower application execution costs, IoT device energy consumption, and application response times. Our offloading method will consider the communication overhead, computation time, and resource utilization efficiency by using the Horse Herd Optimization Algorithm (HHOA).

3. System Model

In this stage, we focus on the practical implementation of our approach using real-world data obtained from Kaggle. This dataset is highly relevant for researchers and developers working on algorithm design for cloud–fog computing environments.

The dataset was generated based on a hierarchical architecture comprising a total of thirteen nodes, comprising three cloud nodes and ten fog nodes. This layered architecture mirrors real-world deployment scenarios, where fog nodes serve as intermediate processing units between resource-constrained IoT devices and more powerful cloud servers.

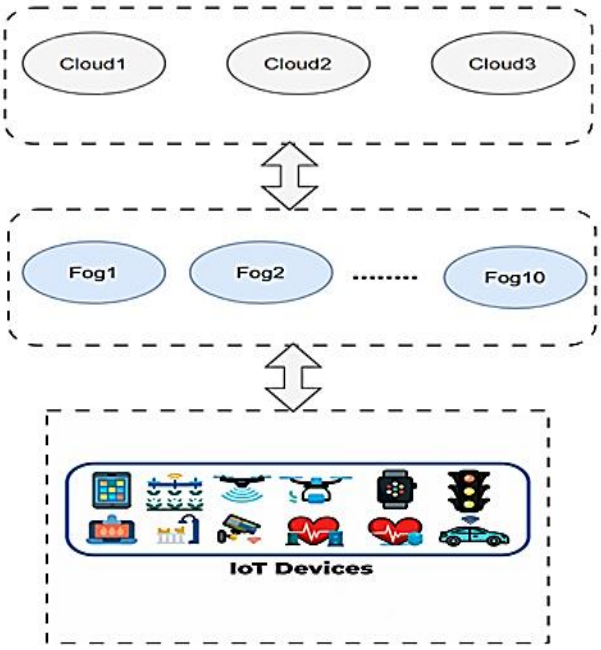

Figure 2. System architecture

Task_Details: This table provides detailed specifications of the tasks, including the number of instructions (expressed in billions, representing CPU processing requirements), memory requirements (in MB), input file size (in MB), and output file size (in MB). These characteristics define the computational workload of each task as table1.

Table 1. Characteristics of IoT Tasks

Property	Value
Number of instructions (10^9)	[1]
Memory required (MB)	[50,200]
Input file size (MB)	[10,100]
Output file size	[10,100]

Node_Details: This table lists the configuration of the nodes, identifying Node1 to Node3 as cloud nodes and Node4 to Node13 as nodes of fog. Every node differs in processing capacity and resource usage cost. The processing power is shown by the MIPS (Million Instructions Per Second) metric. Each node also has costs for bandwidth, memory, and CPU use consumption, as detailed in table2.

Table 2: Resource Parameters for Fog and Cloud Nodes

Parameter	Fog Nodes	Cloud Nodes
CPU rate (MIPS)	[500,1500]	[3000,5000]
CPU usage cost	[0.1,0.4]	[0.7,1.0]
Memory usage cost	[0.01,0.03]	[0.02,0.05]
Bandwidth usage cost	[0.01,0.02]	[0.05,0.1]

3.1. Problem Definition

The task offloading challenge in a hybrid cloud fog setting entails allocating a number of IoT tasks to the available computing nodes for the purposes of minimizing execution time and resource cost. The problem can be modeled by using two major matrices:

- The Task Execution Matrix, which calculates task execution time based on processing speed.
- The Cost Matrix, which estimates the total cost incurred from CPU usage, memory usage as well as bandwidth usage.

The primary objective is to shorten the execution time and resource expenses, which represents a fundamental goal in task offloading models within fog and cloud computing environments. The following equations define this:

1. Task Execution Matrix: A two-dimensional matrix, where the rows represent between tasks and the columns represent nodes. Each cell represents the estimated execution time for a task-node pair. The total execution time of a node N_i , $\mathrm{EXT}(N_i)$, is the sum of the execution times of all tasks allocated to the node. In equations:

$$\mathrm{EXT}(N_i) = \sum_{T_k^i \in N_{i-}\ \mathrm{Tasks}} \mathrm{ExeTime}\left(T_k^i\right) = \sum_{T_k \in N_{i-}\ \mathrm{Tasks}} \frac{\mathrm{length}(T_k)}{\mathrm{CPUrate}(N_i)} \quad (1)$$

2. Cost Matrix: is a two-dimensional array of values indicating the rows considered as tasks and the columns considered to be nodes. Each value represents the total execution cost for a task in the cloud/fog execution model when a task is executed on a node.

When a task $T_k$is executed on node $N_i$the total cost is computed by summing the following three components:

$$\mathrm{Cost}\left(T_k^i\right) = c_p\left(T_k^i\right) + c_m\left(T_k^i\right) + c_b\left(T_k^i\right) \quad (2)$$

The sum of the costs for every task in the system is determined as follows:

The sum of the costs for every task in the system is determined as follows:

$$\mathrm{TotalCost} = \sum_{T_k^i \in\ \mathrm{NodeTasks}} \mathrm{Cost}\left(T_k^i\right) \quad (3)$$

3.2. Task Offloading Optimization

This paper focuses on a hook task offloading method with Horse Herd management Algorithm (HHOA) use in fog computing. This approach aims to reduce task

reaction time and minimize total costs relating to CPU, memory, and bandwidth consumption

Table3. HHOA Algorithm Parameters

Parameter	Value
Number of Tasks	120
Number of Nodes	13
Population Size	50
Number of Iterations	200
Dimension	120
Lower Bound	1
Upper Bound	13
Stallion Ratio	0.2
Colt Ratio	0.13
Number of Stallions	10
Number of Foals	40

The Horse Herd enhancing Algorithm (HHOA) will be used to mimic the fog environment during this phase because of the hybrid cloud's workload offloading issue. The enculturation approach of the HHOA takes on their social hierarchy and individual movement to reproduce the acceptable intelligent collective action of horses in a herd. Each horse in this simulation represents a possible solution of the tasks allocated to one of the available nodes and fog or cloud. A position vector in a d-dimensional space is used to represent each solution. The total number of tasks are indicated by d.

$$X_i = [x_{i1}, x_{i2}, \dots, x_{id}] \quad (4)$$

Here, X_i illustrates the position of the individual i-th horse and each component x_{ij} indicates the task assigned j to a specific node in the infrastructure.

The optimization process involves both exploitative and explorative behaviors that allow the population to converge toward an optimal assignment. The elite individuals in the population, known as stallions, undergo local exploitation. Their movement is defined by:

$$X_i^{t+1} = X_i^t + r_1 \cdot (X_{\text{best}}^t - X_i^t) + r_2 \cdot (X_j^t - X_i^t) \quad (5)$$

where r_1 and r_2 are random numbers in the range of values $[0,1]$, X_{best}^t is the current best-known solution, and X_j^t is a randomly selected horse used to introduce diversity into the update.

The remaining individuals in the population, referred to as foals, exhibit random movements in the solution space to encourage exploration and prevent premature convergence:

$$X_i^{t+1} = X_i^t + \alpha \cdot \text{randn}(d) \quad (6)$$

where α is a step size control parameter, and $\text{randn}(d)$ is a random vector with d dimensions that is taken from a normal distribution.

To preserve high-quality solutions, the algorithm incorporates a resting strategy that prevents the replacement of superior individuals if no improvement is achieved:

$$X_i^{t+1} = X_i^t \text{ if } f(X_i^t) < f(X_i^{t-1}) \quad (7)$$

Each solution is evaluated using a multi-objective fitness function that considers three key criteria:

$$f(X) = w_1 \cdot \text{Total Cost} + w_2 \cdot \text{Makespan} + w_3 \cdot \text{Energy Consumption} \quad (8)$$

where:

- Total Cost is the sum of execution, memory, and bandwidth costs for all tasks,
- Makespan is the maximum completion time across all nodes,
- Energy Consumption quantifies the total energy used by the assigned nodes,
- $w_1, w_2,$ and w_3 are weighting factors that can be tuned to emphasize specific objectives.

The algorithm continues iterating through a fixed number of generations or until a convergence criterion is met. Throughout this process, horses adaptively adjust their positions based on the behavioral operators described above, progressively moving the herd toward an optimal or near-optimal task allocation configuration. The pseudocode for the standard Horse Herd Optimization Algorithm (HHOA) is presented in Algorithm 1.

Algorithm 1: Psedocode for the standard HHOA for The Task Offloading Optimization

Generate an initial population of N horses, $\hat{X}_{t_2}$ $(i \in N)$,
Set HHOA parameters: population size N_s number of stallions N_{stallion}, number of foals, and max iterations $t_{\max}$
For each horse X_{i_y} evaluate the fitness function:
$f(X) = w_1\text{-TotalCost} + w_2\,\text{Makespan} + w_3\text{EnergyConsumption}$
$t \leftarrow 0$
while $t < t_{\max}$ **do**
  Identify the best solution in the population, X_{best}
  for each horse X_i in population **do**
    if X_i is a stallion then
      Select as other stallion $X_{\bar{j}}$ randomly
      Update X_i using:
        $X_i \leftarrow X_i + r_{1\text{-}}(X_{\text{best}} - X_i) + r_{2\text{-}}(X_{j-} - X_i)$
    else if X_i is a ***foal*** then
    Update X_i using:
      $X_i \leftarrow X_i + \alpha\text{-}randn(d)$, then $X_i \leftarrow X_{j\,old}$
    Apply resting mechanism: if $(X_{(new)}) \geq f$ [nodes]
  Evaluate $f(X)$
Return $X_{\text{best}}(t)$ as the optimal task-node allocation

Figure 3-1. illustrates the Horse Herd Optimization Algorithm (HHOA) as applied to task offloading in a distributed computing environment.

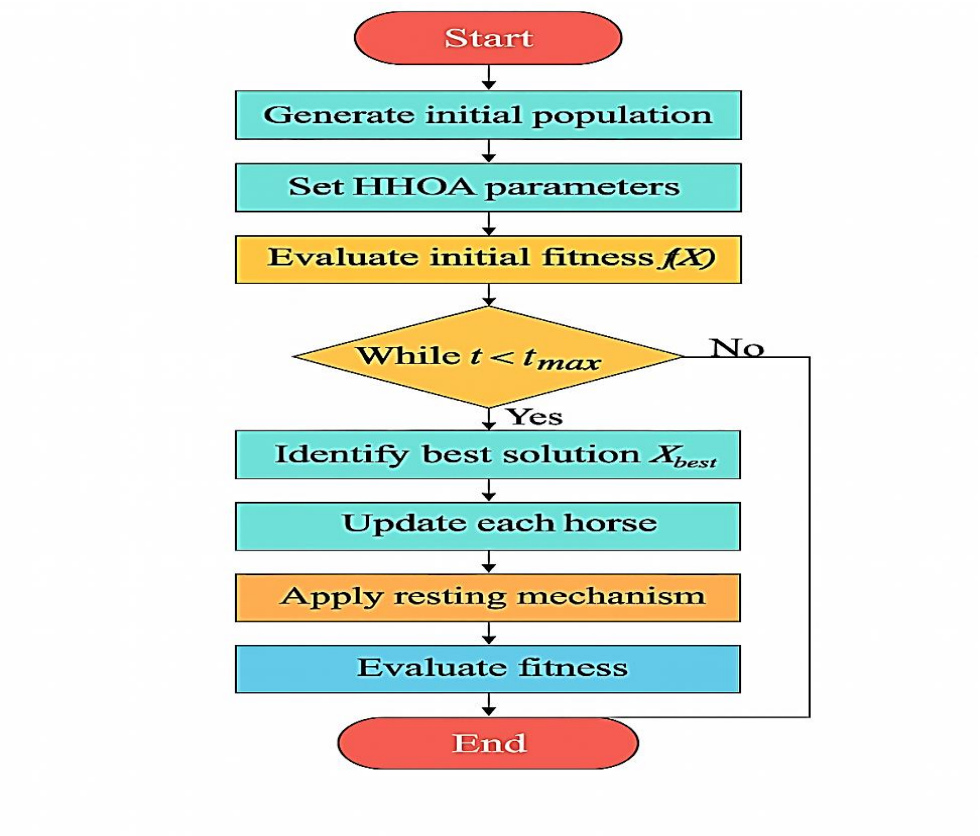

Figure 3-2. Horse Herd Optimization Algorithm (HHOA) for Task Offloading

4. Evaluation System

Results from simulations show how well the suggested Horse Herd Efficiency Algorithm (HHOA) carries out offloading tasks in a fog-cloud hybrid setting. The algorithm's whole execution time was 0.9731 seconds, resulting in an optimal scheduling cost of 2.155. Makespan, the maximum task completion time, was 6.1296 seconds. These results validate that, as indicated in Table 4, the algorithm provides effective scheduling with respect to execution time and cost.

Table 4. Execution Times and Costs of First 10 Tasks

Task	Execution Time (s)	Execution Cost
1	0.8453	0.3349
2	0.2832	0.8712
3	0.2852	0.8046
4	0.2832	0.8455

5	0.8628	0.3509
6	0.8460	0.3307
7	0.8757	0.3274
8	0.2852	0.8467
9	0.8628	0.3371
10	0.8460	0.3463

Figure 4 illustrates the optimization progress of the HHOA algorithm over 200 iterations. The best fitness value drops rapidly at the beginning, indicating fast convergence, and stabilizes after around 150 iterations. This shows how well the algorithm reduces task offloading costs in a hybrid fog, cloud environment.

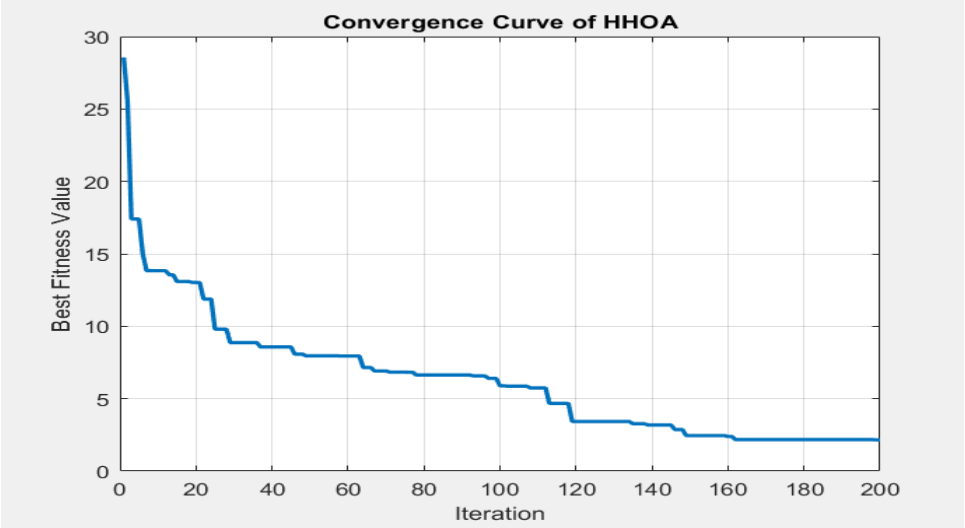

Figure 4. Convergence Curve of the HHOA Algorithm

The results indicate, that as expected, the majority of tasks were sent to cloud nodes due to the faster processing capabilities. Also, importantly, even though the costs of using cloud resources was higher, the HHOA algorithm did attribute smaller task executions to cloud resource in order to thereby balance cost with performance. Furthermore, fog nodes (which have limited resource) were executing certain longer task executions which helped ensure that their resources were available for executions that can take a significant amount of time, and avoid overload as shown in figure 5.

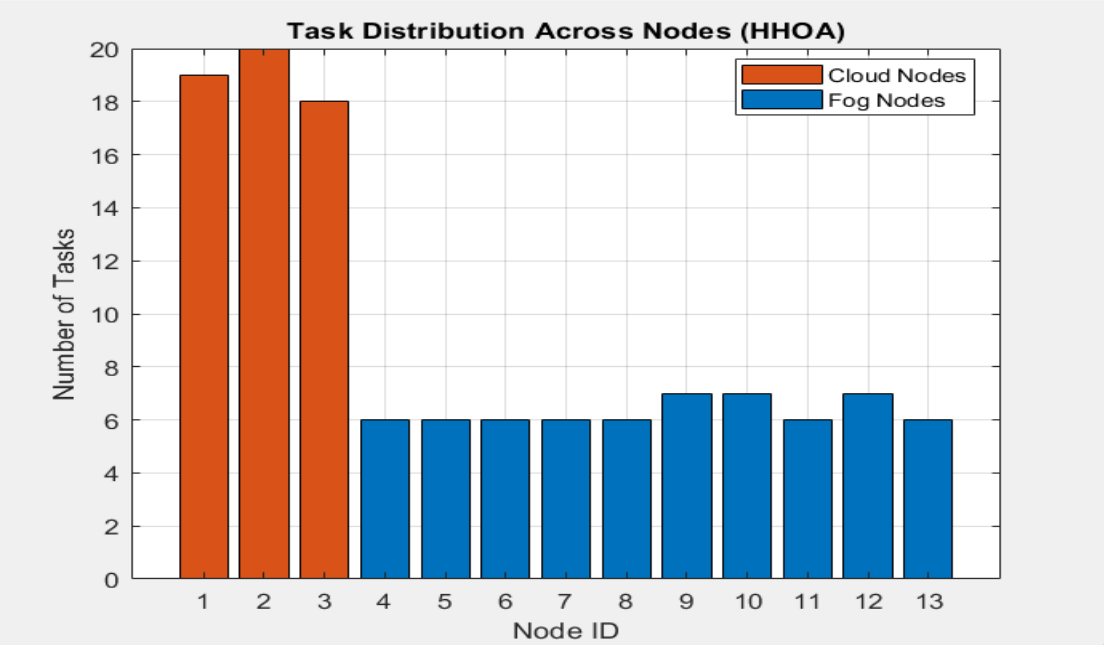

Figure 5. Task Distribution Across Nodes (HHOA)

In task offloading systems, execution time is one of the major factors for assessing performance and responsiveness when using IoT applications in real-time. Due to the considerable computing disparity between cloud and fog nodes, the execution time of tasks can significantly differ depending on the node type assigned to the task. Moreover, knowing how tasks are allocated among nodes and the execution timethey incur is an important measure for determining whether or not the optimization strategies (ex. Horse Herd Optimization Algorithm (HHOA) and/or other) was successful. The subsequent figure compares the execution time per task to identify performance differences between fog and cloud computing environments. Figure 6 shows the execution times of assignments given to cloud and fog nodes. Assignments made to cloud nodes show optimal average and consistent execution times of around 0.28 seconds, as cloud infrastructure has better processing power. On the other hand, tasks assigned to fog nodes show high and variable execution time with an average time between 0.84 to 0.91 seconds. While cloud resources are much more expensive than fog resources, the HHOA (Horse Herd Optimization Algorithm) selected cloud nodes for tasks that required quick execution, trading off speed and cost. The average execution time for all tasks is around 0.6 seconds, which shows that HHOA is a valuable objective to optimizing task allocation in a hybrid fog and cloud infrastructure.

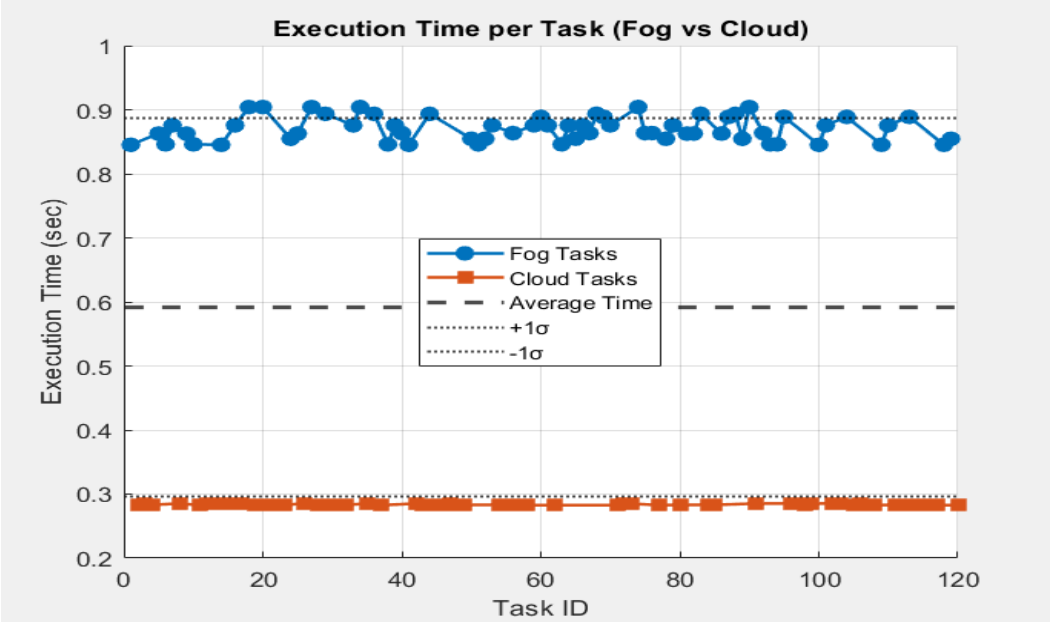

Figure 6. Execution Time per Task

Additionally, figure 7 highlights the execution cost across tasks split between fog and cloud nodes. As shown, cloud nodes have much higher solidified execution costs of between 0.7 and 0.9 as they are given the benefit of superior computational capabilities compared with fog nodes. Although cloud nodes do have higher execution costs, the HHOA algorithm evaluates the cost relative to the time-sensitive tasks assigned to cloud nodes to provide shorter execution times, offsetting the higher costs. Meanwhile, fog nodes experience lower execution costs, approximately 0.33 to 0.36, with more stability than cloud nodes and are better used with tasks that were not as time-critical. The HHOA algorithm is still able to demonstrate the ability to balance performance and cost through distribution that both plague the performing times for each assigned node through using fog and cloud layers for task offloading.

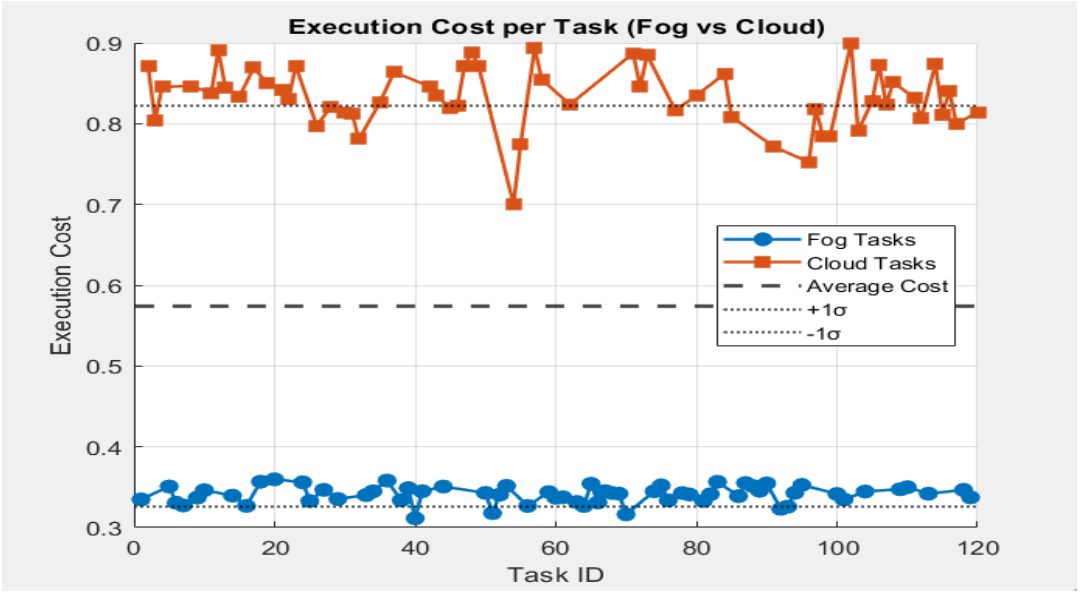

Figure 7. Execution Cost per Task

Resource consumption is analyzed across nodes in the HFC system. The results show that cloud nodes typically have higher usage of both CPU and memory compared to fog nodes, as seen in Figures 8 and 9. This is because the HHOA algorithm mapped higher CPU- and memory-intense tasks, or more time-constrained tasks, to the more powerful processing capabilities of cloud nodes. On the other hand, fog nodes exhibited a more equal utilization distribution. It made sense for fog nodes to handle light tasks that adhered to the resource constraints and were also usually closer to end devices for decreased latency responses. This distribution of tasks demonstrates the effectiveness of the HHOA algorithm to enhance overall performance, and ensure resource wastefulness was avoided in cloud and fog nodes.

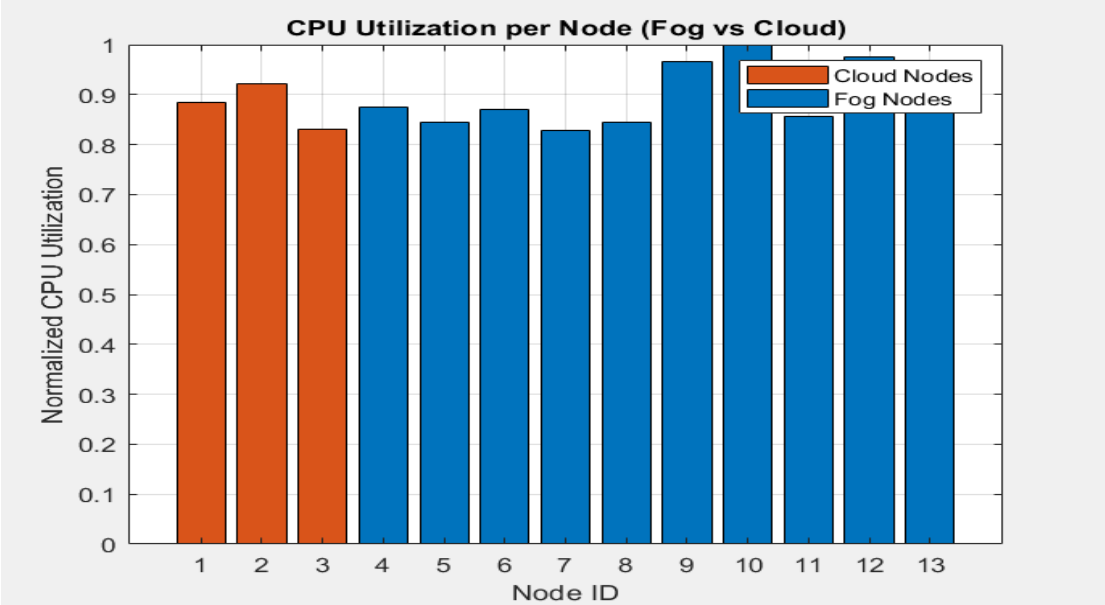

Figure 8. CPU Utilization per Node

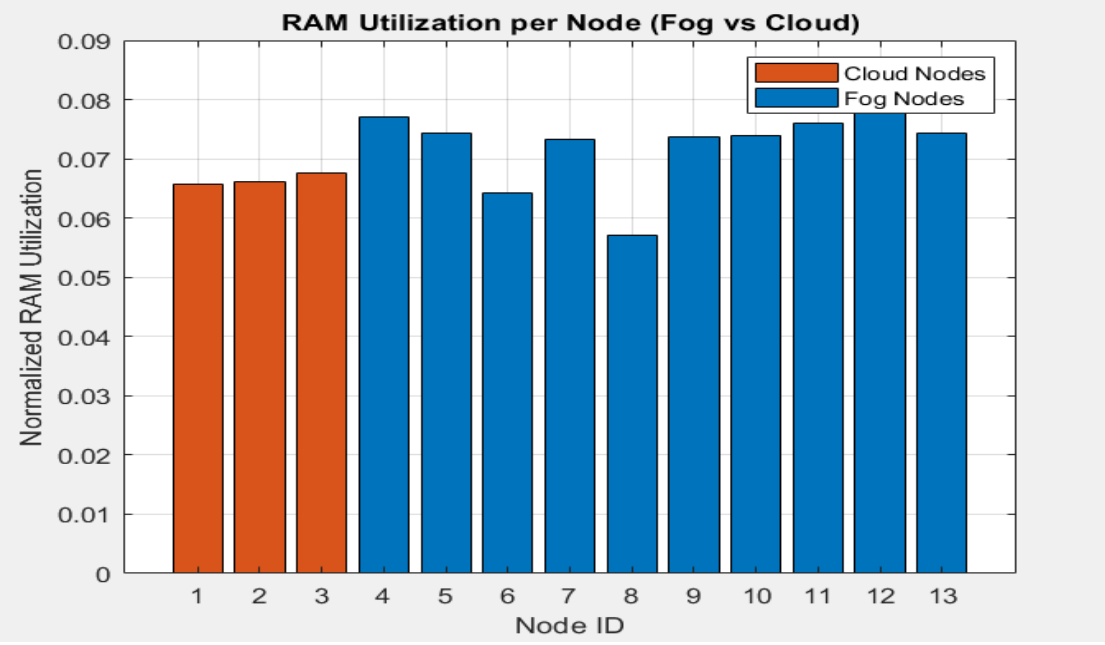

Figure 9. RAM Utilization per Node

Energy consumption plays an essential role in evaluating efficiency in hybrid fog and cloud systems. Energy consumption affects both operating costs and sustainability in a way that all parties can relate. In this review, cloud nodes will exhibit much greater energy consumption than fog nodes due to the higher processing power and procedures requiring greater resources of cloud nodes. Fog nodes, will be put under substantially lower workloads compared to cloud nodes, resulting in a sizeable decrease in the fog node's energy consumption. This better aligns fog nodes with applications that are energy sensitive. The Horse Herd Optimization Algorithm (HHOA) provides an adequate balance between task assignments on cloud versus fog nodes. This minimizes overall energy consumption while maintaining performance. This task distribution lowers the total energy use and also improves cost efficiency throughout the system.

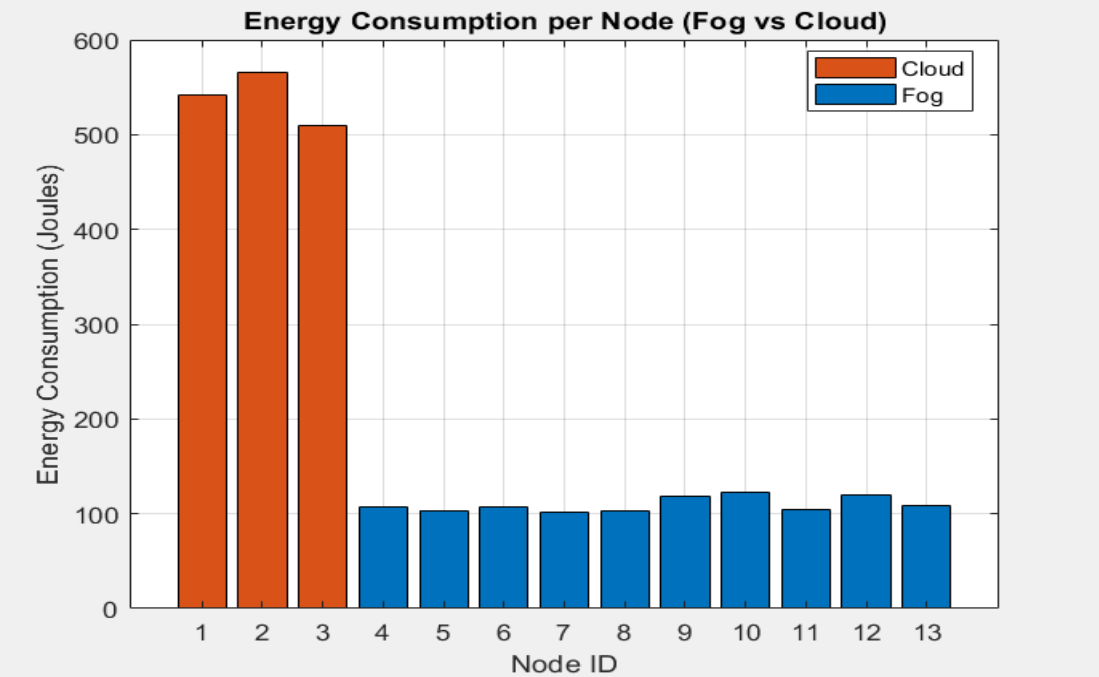

Figure 10. Energy Consumption per Node

5. Conclusion

The study reviewed literature relevant to task offloading in hybrid cloud and fog environments. The Horse Herd Improvement Algorithm (HHOA) was utilized to reduce task execution time and resource cost in IoT networks, which helps to improve task offloading. The algorithm's execution time of 0.9731 seconds is small compared to many metaheuristic algorithms. This shows how quickly it finds the best solution. The also suggested that the algorithm was able to find a low-cost task distribution based on available resources, as it achieved a scheduling cost of 2.155. Additionally, the makespan maximum time to complete tasks, was 6.1296 seconds, which implies there was an even distribution of the work, suggesting all tasks completed in a reasonable amount of time. As per the analysis of execution time and cost per task, the longest execution time was 0.8757 seconds and the highest execution cost was 0.8712, meaning that the task allocation was well-balanced, as it indicates that there was a trivial difference between tasks. The HHOA algorithm was able to efficiently assign time-critical tasks to the cloud nodes since they have more processing

capability and used this to their advantage. Fog nodes were assigned a less time-critical task in order to conserve their limited amount of CPU, RAM, and power. Overall, this method effectively utilized both fog and cloud layers in hybrid IoT environments, while meeting the need to balance cost versus performance. A significant benefit of this work was its use of a real-world IoT fog-cloud dataset from Kaggle to evaluate the algorithm. In contrast, most previous HHOA studies used randomly generated datasets. This practical method more closely resembles real-world deployment scenarios and yields a more realistic outcome. Additionally, it increases the results' dependability and relevance.

6. Future Work

Future research may use this work as a foundation by exploring the Horse Herd Improvement Algorithm (HHOA) across various real-world datasets from distinct IoT environments, including intelligent transportation systems (ITSs), smart cities, and health monitoring IoT applications. This can validate the strength of the HHOA to sustain a variety of workloads, network architecture, and resource constraints. Additionally, the scalability and applicability of HHOA can be measured by adding dynamic task arrivals in real time and comparing the HHOA performance to other metaheuristic and machine learning solutions.

References

[1] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. 2015 Workshop on Mobile Big Data (Mobidata '15)*, New York, NY, USA: ACM, 2015, pp. 37–42. [Online]. Available: http://doi.acm.org/10.1145/2757384.2757397

[2] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *2012 Proc. IEEE INFOCOM*, Mar. 2012, pp. 945–953.

[3] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: A computation offloading framework for smartphones," in M. Gris and G. Yang, Eds., *Mobile Computing, Applications, and Services*, Berlin, Heidelberg: Springer, 2012, pp. 59–79.

[4] E. Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *Proc. 8th Int. Conf. Mobile Systems, Applications, and Services (MobiSys '10)*, New York, NY, USA: ACM, 2010, pp. 49–62. [Online]. Available: http://doi.acm.org/10.1145/1814433.1814441

[5] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the Internet of Things," in *Proc. 2nd ACM SIGCOMM Workshop on Mobile Cloud Computing (MCC '13)*, New York, NY, USA: ACM, 2013, pp. 15–20. [Online]. Available: http://doi.acm.org/10.1145/2491266.2491270

[6] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.

[7] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017. DOI: 10.1109/JIOT.2017.2750180

[8] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *First Edition of the MCC Workshop on Mobile Cloud Computing*, 2012, pp. 13–16.

[9] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, 2016. DOI: 10.1109/TVT.2015.2478060

[10] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *J. Syst. Archit.*, vol. 98, pp. 289–330, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1383762118306349

[11] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma, "Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration," *IEEE Trans. Comput.*, vol. 67, no. 9, pp. 1287–1300, 2018. DOI: 10.1109/TC.2017.2776363

[12] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource

allocation for cloud-assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, 2019. DOI: 10.1109/TVT.2019.2920459

[13] T. T. Nguyen, L. B. Le, and Q. Le-Trung, "Computation offloading in MIMO-based mobile edge computing systems under perfect and imperfect CSI estimation," *IEEE Trans. Serv. Comput.*, vol. 14, no. 6, pp. 2011–2025, 2019. DOI: 10.1109/TSC.2018.2837891

[14] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12313–12325, 2018. DOI: 10.1109/TVT.2018.2878138

[15] I. Ullah, H.-K. Lim, Y.-J. Seok, and Y.-H. Han, "Optimal task offloading with deep Q-network for edge-cloud computing environment," in *2022 13th Int. Conf. Inf. Commun. Technol. Convergence (ICTC)*, IEEE, 2022, pp. 406–411. DOI: 10.1109/ICTC55334.2022.9950965

[16] A. Robles-Enciso and A. F. Skarmeta, "A multi-layer guided reinforcement learning-based tasks offloading in edge computing," *Computer Networks*, vol. 220, p. 109476, 2023. DOI: 10.1016/j.comnet.2022.109476

[17] T. H. Binh, H. Vo, B. M. Nguyen, and H. T. T. Binh, "Reinforcement learning for optimizing delay-sensitive task offloading in vehicular edge–cloud computing," *IEEE Internet of Things Journal*, vol. 11, no. 2, pp. 2058–2069, 2023. DOI: 10.1109/JIOT.2023.3236115

[18] A. Bhattacharya and P. De, "A survey of adaptation techniques in computation offloading," *J. Netw. Comput. Appl.*, vol. 78, pp. 97–115, Jan. 2017. [Online]. Available: https://doi.org/10.1016/j.jnca.2016.10.023

[19] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mob. Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb. 2013. [Online]. Available: http://dx.doi.org/10.1007/s11036-012-0368-0

[20] A. Yousefpour, G. Ishigaki, and J. P. Jue, "Fog computing: Towards minimizing delay in the Internet of Things," in *2017 IEEE Int. Conf. Edge Computing (EDGE)*, June 2017, pp. 17–24. DOI: 10.1109/EDGE.2017.13

[21] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 1, pp. 108–119, Jan. 2017. DOI: 10.1109/TETC.2015.2494342

[22] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016. DOI: 10.1109/JIOT.2016.2597162

[23] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Trans. Comput.*, vol. 65, no. 12, pp. 3702–3712, Dec. 2016. DOI: 10.1109/TC.2016.2604341

[24] Y. Chen, Y. Chang, C. Chen, Y. Lin, J. Chen, and Y. Chang, "Cloudfog computing for information-centric Internet-of-Things applications," in *2017 Int. Conf. Appl. System Innovation (ICASI)*, May 2017, pp. 637–640. DOI: 10.1109/ICASI.2017.7988328

[25] Y.-L. Jiang, Y.-S. Chen, S.-W. Yang, and C.-H. Wu, "Energy-efficient task offloading for time-sensitive applications in fog computing," *IEEE Syst. J.*, vol. 99, pp. 1–12, Nov. 2018. DOI: 10.1109/JSYST.2018.2881638

[26] M. Huang, W. Liu, T. Wang, A. Liu, and S. Zhang, "A cloud-MEC collaborative task offloading scheme with service orchestration," *IEEE Internet of Things Journal*, 2019. DOI: 10.1109/JIOT.2019.2920452

[27] Y. Liu, Z. Zeng, X. Liu, X. Zhu, and M. Z. A. Bhuiyan, "A novel load balancing and low response delay framework for edge-cloud network based on SDN," *IEEE Internet of Things Journal*, 2019. DOI: 10.1109/JIOT.2019.2902824

[28] S. Zahoor, N. Javaid, A. Khan, B. Ruqia, F. J. Muhammad, and M. Zahid, "A cloud-fog-based smart grid model for efficient resource utilization," in *2018 14th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, June 2018, pp. 1154–1160. DOI: 10.1109/IWCMC.2018.8450487

[29] S. Zahoor, S. Javaid, N. Javaid, M. Ashraf, F. Ishmanov, and M. K. Afzal, "Cloud–fog–based smart grid model for efficient resource management," *Sustainability*, vol. 10, no. 6, 2018. [Online]. Available: https://www.mdpi.com/2071-1050/10/6/2079

[30] S. A. A. Naqvi, N. Javaid, H. Butt, M. B. Kamal, A. Hamza, and M. Kashif, "Metaheuristic optimization technique for load balancing in cloud-fog environment integrated with smart grid," in L. Barolli, N. Kryvinska, T. Enokido, and M. Takizawa, Eds., *Advances in Network-Based Information Systems*, Cham: Springer, 2019, pp. 700–711. DOI: 10.1007/978-3-030-15809-8_59

[31] S. Ningning, G. Chao, A. Xingshuo, and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning," *China Commun.*, vol. 13, no. 3, pp. 156–164, Mar. 2016. DOI: 10.1109/CC.2016.7447795

[32] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterprise Inf. Syst.*, vol. 12, no. 4, pp. 373–397, 2018. DOI: 10.1080/17517575.2017.1304579

[33] H. T. T. Binh, T. T. Anh, D. B. Son, P. A. Duc, and B. M. Nguyen, "An evolutionary algorithm for solving task scheduling problem in cloud-fog computing environment," in *Proc. 9th Int. Symp. Inf. Commun. Technol. (SoICT 2018)*, New York, NY, USA: ACM, 2018, pp. 397–404. [Online]. Available: http://doi.acm.org/10.1145/3287921.3287984

[34] B. M. Nguyen, H. T. T. Binh, T. The Anh, and D. B. Son, "Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment," *Appl. Sci.*, vol. 9, no. 9, p. 1730, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/9/1730

[35] A. Mebrek, L. Merghem-Boulahia, and M. Esseghir, "Efficient green solution for a balanced energy consumption and delay in the IoT-fog-cloud computing," in *2017 IEEE 16th Int. Symp. Netw. Comput. Appl. (NCA)*, Oct. 2017, pp. 1–4. DOI: 10.1109/NCA.2017.8080318

[36] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *2018 IEEE Int. Conf. Edge Computing (EDGE)*, Los Alamitos, CA, USA: IEEE, Jul. 2018, pp. 66–73. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/EDGE.2018.00016

[37] Q. Wang and S. Chen, "Latency-minimum offloading decision and resource allocation for fog-enabled Internet of Things networks," *Trans. Emerg. Telecommun. Technol.*, p. e3880, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3880

[38] C. Canali and R. Lancellotti, "GASP: Genetic algorithms for service placement in fog computing systems," *Algorithms*, vol. 12, no. 10, p. 201, 2019. [Online]. Available: https://www.mdpi.com/1999-4893/12/10/201