Journal of Theoretical and Applied Information Technology
15th June 2025. Vol.103. No.11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

OPTIMIZED TASK SCHEDULING IN FOG-CLOUD ENVIRONMENTS USING A COST-AWARE GENETIC ALGORITHM

YOUSSEF OUKISSOU [1], HAMZA ELHAOU [2], DRISS AIT OMAR [3], HICHAM ZOUGAGH [4], SAMIR ELOUAHAM [5]

[1]Information Processing and Decision Support Laboratory, University of Sultan Moulay Slimane, Faculty of Science and Technologies, Beni Mellal 23500, Morocco
[2]Information Processing and Decision Support Laboratory, University of Sultan Moulay Slimane, Faculty of Science and Technologies, Beni Mellal 23500, Morocco
[3]Information Processing and Decision Support Laboratory, University of Sultan Moulay Slimane, Faculty of Science and Technologies, Beni Mellal 23500, Morocco
[4]Information Processing and Decision Support Laboratory, University of Sultan Moulay Slimane, Faculty of Science and Technologies, Beni Mellal 23500, Morocco
[5]LIST Laboratory, National School of Applied Sciences, Ibn zohr university, Agadir, Morocco

E-mail: [1]youssef.oukissou@usms.ma, [2]hamza.elhaou@usms.ma, [3]d.aitomar@usms.ma, [4]h.zougagh@usms.ma, [5]elouahamsamir@gmail.com

ABSTRACT

Cloud technology offers flexible computing and storage solutions over the internet. However, for latency-sensitive applications such as smart healthcare and smart cities, the reliance on centralized cloud data centers leads to significant performance issues, particularly in terms of delay. Fog and edge computing paradigms aim to address these issues by bringing resources closer to end-users, thus reducing latency and enhancing energy efficiency. In this paper, we propose a cost-aware, genetic-based (CAG) task scheduling algorithm tailored for fog-cloud environments, which seeks to improve cost efficiency for real-time applications with strict deadlines. Our approach is implemented and evaluated using the PureEdgeSim simulator, focusing on key performance metrics such as latency, network congestion, and cost. The results demonstrate that the proposed algorithm surpasses existing techniques like Round-Robin and Trade-off algorithms, achieving superior performance in terms of cost and throughput efficiency.

These results demonstrate that the CAG algorithm is an effective solution for task scheduling in fog-cloud environments, offering better cost and deadline management compared to existing methods. This research has significant implications for latency-sensitive IoT applications, such as smart healthcare and smart cities, where efficient resource allocation and cost optimization are critical.

Keywords: *Fog Computing, IoT Scheduling, Edge Computing, Resource Allocation, Task Management*

1. INTRODUCTION

The Internet of Things (IoT) [1] represents a major technological shift in our digital age. This vast network connects billions of smart objects and sensors to the Internet, enabling the real-time collection and exchange of massive data about our physical environment [2], IoT applications are ubiquitous, spanning various fields such as smart cities, transportation, healthcare, Industry 4.0, home automation, precision agriculture, and more. According to Gartner, the number of connected IoT devices is expected to reach 25 billion by 2025, generating an enormous data traffic estimated at 175 zettabytes per year \cite{bonomi2012fog} This explosion of IoT data poses significant challenges in terms of network infrastructure, bandwidth, storage capacity, and especially the computational power needed to process and analyze these data streams in a timely manner.

Initially, cloud computing was seen as the ideal solution to host IoT applications and provide on-demand elastic computing and storage resources.

However, the inherently centralized nature of the cloud makes it unsuitable for meeting the critical requirements of latency, responsiveness, privacy, and service continuity demanded by many strategic IoT applications like smart healthcare and smart cities [3].

To address these limitations, new decentralized paradigms of proximity computing, known as fog computing and edge computing, have recently emerged. Their principle is to deploy computing, storage, and networking capabilities at the edge of the cloud, closer to IoT devices, thus enabling data processing at the source, in real time, and in a secure manner [4], by intelligently and complementarily combining fog, edge, and cloud computing, it becomes possible to leverage the advantages of each layer [5].

However, effectively orchestrating heterogeneous resources across this distributed hybrid architecture presents numerous challenges, particularly regarding the efficient scheduling of IoT tasks and data flows. It is essential to develop scheduling strategies that consider the multiple constraints inherent to various IoT applications (deadlines, quality of service, energy consumption, operational costs, etc.) to fully exploit the benefits offered by the fog-edge-cloud computing continuum [6].

In this paper, we propose a Cost-Aware Genetic-based (CAG) task scheduling algorithm, designed for fog-cloud environments. The algorithm aims to improve cost efficiency, reduce latency, and minimize network congestion in real-time applications with strict deadlines. We evaluate our approach using the PureEdgeSim simulator, focusing on key performance metrics such as latency, cost, and network congestion.

The structure of this paper is organized as follows: Section 2 discusses the background and related works in task scheduling techniques, followed by the proposed methodology in Section 3. Section 4 presents the simulation settings and results, and finally, Section 5 concludes the paper with future research directions.

2. BACKGROUND AND RELATED WORKS

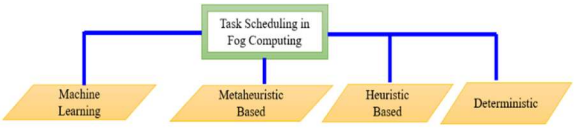

Figure 1: The Classification Of Task Scheduling Techniques In Fog Computing.

Figure 1 presents an overview of task scheduling techniques in fog environments, organizing them into four main categories: machine learning, heuristic-based approaches, metaheuristic-based methods, and deterministic strategies. The articles analyzed in this study are placed within these categories to highlight their relevance. In the sections that follow, we review and compare these works, focusing on qualitative parameters to provide a comprehensive understanding.

2.1 An Overview of Machine Learning Mechanisms

Conventional approaches are ineffective with large data sets [7], [8], Machine learning techniques play a crucial role in improving the efficiency and precision of fog task scheduling. By examining historical data and uncovering patterns, these algorithms can forecast future traffic loads and strategically optimize task allocation to meet system demands effectively. This approach helps minimize latency, enhance overall system performance, and maximize resource utilization. Furthermore, machine learning algorithms can identify anomalies or potential system failures early on, enabling proactive measures to mitigate issues before they escalate. Incorporating machine learning into fog task scheduling can greatly improve the efficiency of the scheduling process while boosting the overall performance of fog computing systems.

In this section, the chosen machine learning-based techniques are grouped into three distinct categories. The strengths and limitations of these methods are thoroughly examined and discussed.

2.1.1 Artificial Neural Network (ANN) Based Methods for Task Scheduling

Arri, al [9] designed an enhanced system for managing Task Group Aggregation (TGA) overflow in fog computing environments, leveraging neural computing techniques. They developed an Artificial Neural Network (ANN) based algorithm that detects overloaded servers and efficiently redistributes the

data to Virtual Machines (VMs) within the fog computing network. Various factors such as CPU, memory, and bandwidth are considered to balance VMs. Additionally, the Artificial Bee Colony (ABC) algorithm is used to separate services and users based on their specific quality requirements. The introduced ANN-based overflow handling algorithm improves response time and success rate parameters compared to current approaches. However, energy consumption is not addressed in this paper. Considering the computational complexity of ANN algorithms, future work could extend this research by optimizing the models' computational complexity and incorporating energy efficiency considerations into the algorithm.

2.1.2 Deep Reinforcement Learning (DRL) Based Methods for Task Scheduling

DRL-based methods offer a promising approach to improving task scheduling in fog computing environments, with the potential to reduce energy consumption and enhance overall system performance. These methods learn from past experiences and make decisions based on real-time data, leading to reduced latency. However, a potential drawback is the complexity and computational cost of implementing DRL algorithms, which may require significant resources.

For instance, Li, al [10] utilized deep learning models for fog task scheduling in mobile crowd sensing applications. The authors enhanced the efficiency of image data processing by deploying IoT devices as part of a multilayer structure in a deep learning Convolutional Neural Network (CNN) model. The simulation results showed that the proposed solution minimizes bandwidth costs from fog nodes to the cloud layer, thereby reducing cloud layer computations. This approach focuses on optimizing energy consumption and reducing delay by dynamically adjusting the sampling rate of sensor data. It also enables proactive measures to prevent potential failures by predicting future resource availability and adjusting task scheduling accordingly. This helps meet execution deadlines and improves response times. However, as the system scale grows, the proposed model struggles to converge, resulting in unstable outcomes.

Gazori, al [11] also employed a DRL-based model for fog task scheduling in mobile crowd sensing applications. They proposed an efficient task scheduling algorithm based on a double deep Q-learning model. The purpose of this algorithm is to reduce service latency and measurement costs within the constraints of resources and time limits. According to the results presented in the paper, the model outperforms simpler techniques in terms of delay and task completion time. Additionally, the proposed algorithm addresses single-point failure problems and fog load balancing issues. However, the algorithm reaches a solution at a slow pace and faces uncertainty in high-dimensional state-action spaces. The article does not provide specific information on how the DRL approach performs in terms of network bandwidth and energy consumption criteria.

Swarup, al [12] and [13] proposed an algorithm called Clipped Double Deep Q-learning for IoT task scheduling in fog-based environments using deep reinforcement learning to address service latency and energy efficiency. The proposed model utilizes experience replay and target network techniques. The authors aim to improve energy efficiency, reduce costs, and minimize delays in task scheduling. The algorithm clusters IoT devices based on their resource requirements and schedules tasks for each cluster using deep reinforcement learning. A parallel queue is also employed to ensure optimal resource utilization without lag. However, potential issues with execution time and network bandwidth may arise.

2.1.3 Graph-Based Deep Learning Methods for Task Scheduling

Graph Neural Networks (GNNs) are a type of neural network tailored for working with graph-structured data. This approach has attracted considerable interest in recent years because of its effectiveness in tackling challenges across various fields, including social network analysis and privacy-preserving machine learning. Recently, GNNs have emerged as a promising solution for optimizing task scheduling in fog computing by leveraging the power of deep learning and graph theory.

Graph-Based Deep Learning Methods use graph-based models to represent the relationships between tasks and resources in the fog computing environment. These models can capture complex dependencies between tasks and resources, allowing for more accurate scheduling decisions. However, these methods can be computationally expensive and require large amounts of data to train.

For instance, Jamil, al [14] proposed a new algorithm called IRATS for online resource allocation and task scheduling in a vehicular fog

network. The algorithm, based on deep reinforcement learning, considers both priority and deadline constraints of tasks. It first predicts future resource requirements using a Long Short-Term Memory (LSTM) network. Then, a DRL agent allocates resources to tasks based on their priority and deadline constraints while considering the current state of the network. The DRL agent learns from past experiences to make better decisions in the future. IRATS can handle dynamic changes in the network, prioritize tasks based on their importance, and meet task deadlines. However, training the DRL agent requires significant computational resources, and the accuracy of the LSTM network's predictions may be affected by changes in the network environment.

Furthermore, Li, al [15] developed a deep-graph-based reinforcement learning approach for joint cruise control and task offloading in aerial EdgeIoT systems. This approach uses a GNN to model the complex relationships among system components and optimize joint control and offloading decisions. The proposed method improves system performance, reduces energy consumption, and enhances reliability. However, it requires significant computational resources and may suffer from scalability issues in large-scale systems. Overall, the paper presents a promising approach for optimizing EdgeIoT systems, but further research is needed to address its limitations.

Table 1: Summary Of The Task Scheduling In Machine Learning Mechanisms.

Journal	Technique	Advantage	Weakness
Arri, et al. (2021)	ANN-based overflow management algorithm	Low response time	High execution time, High energy consumption
Li, et al. (2019)	DRL-based models for fog task scheduling in a mobile crowd-sensing application	Low latency, Low energy consumption, Low response time, Low execution time	High uncertainty, High cost, High resource utilization, High execution time
Gazori, et al. (2020)	Double deep Q-learning model	Low latency, Low response time, Low execution time	High uncertainty, High cost, High resource utilization, High execution time
Swarup, et al. (2021)	Deep reinforcement learning	Low energy consumption, Low latency, Low cost, Low resource utilization	High execution time, High bandwidth
Jamil, et al. (2023)	Graph-based deep learning model	Low execution time	High resource utilization, High uncertainty
Li, et al. (2022)	Deep-graph-based reinforcement learning	Low energy consumption, Low uncertainty	High resource utilization, Low scalability

2.2 An Overview of Heuristic Based Mechanisms

Heuristic-based mechanisms for fog task scheduling are methods that use rules of thumb or best practices to allocate tasks to devices in a fog computing environment. These mechanisms do not rely on machine learning techniques but rather on simple decision-making rules based on experience or intuition. While heuristic-based mechanisms are simple and easy to implement, they may not always result in optimal task allocation [16]. They also do not adapt to changing conditions or learn from past experiences as machine learning-based approaches do.

The following section reviews several heuristic-based fog scheduling methods that utilize various factors to achieve improved scheduling outcomes.
For instance, Arisdakessian, al [17] presented FoGMatch, an intelligent multi-criteria IoT-Fog scheduling approach that uses game theory to allocate tasks to fog nodes. FoGMatch considers multiple factors such as energy consumption, delay, and resource availability in its decision-making process. The authors compare FoGMatch with other scheduling approaches and show that it outperforms them in terms of task completion time and energy

consumption. The advantage of FoGMatch is its ability to adapt to changing conditions and learn from past experiences, as well as considering multiple criteria in its decision-making process, which can lead to better task allocation. However, the disadvantage is that it relies on complex algorithms and may require more computational resources than simpler heuristic-based mechanisms.

Furthermore, Azizi, al [18] formulated the task scheduling problem with the objective of reducing the overall energy consumption of fog nodes while meeting the various qualitative requirements of IoT tasks. The proposed model aims to minimize deadline violation time. The authors mapped IoT tasks to fog nodes using two semi-greedy algorithms: Priority-aware Semi-Greedy (PSG) and PSG with Multi-start method (PSG-M). According to the results, the proposed algorithms increase the percentage of tasks that meet their deadlines, reduce the total time of deadline violations, and optimize the energy consumption of fog resources and system makespan compared to existing algorithms. The time complexity analysis in the paper suggests that this algorithm is a suitable solution for real-time scheduling of IoT tasks in fog computing systems.

In addition, Hosseini, al [19] presented a dynamic scheduling algorithm based on the Priority Queue, Fuzzy, and Analytical Hierarchy Process (PQFAHP). The PQFAHP algorithm combines multiple priorities and can perform multi-criteria prioritization. The authors implemented dynamic scheduling based on various criteria, including completion time, energy consumption, RAM, and deadline. The outcomes revealed that the multi-criteria PQFAHP outperforms benchmark algorithms in terms of waiting time, delay, service level, response time, number of scheduled tasks, and energy consumption.

Xu, al [20] proposed an algorithm based on adaptive dynamic programming to find the best path for processing data with different priorities at fog nodes. The goal of the proposed algorithm is to reduce time delay and energy consumption for priority-based tasks. Simulation results showed that the proposed algorithm increases efficiency and reliability while reducing power consumption.

Table 2: Summary Of The Task Scheduling In Heuristic-Based Mechanisms.

Journal	Technique	Advantage	Weakness
Arisdakessian, et al. [11]	Intelligent scheduling based on game theory	Low energy consumption, Low execution time	High resource utilization
Azizi, et al. [12]	Semi-greedy algorithm	Low energy consumption, Low execution time, Low complexity	
Hosseini, et al. [13]	Fuzzy and analytical hierarchy process	Low latency, Low response time, Low energy consumption	High resource utilization
Xu, et al. [14]	Dynamic programming	Low energy consumption, Low cost, Low execution time	High resource utilization

2.3 An Overview of Metaheuristic Based Mechanisms

In meta-heuristic algorithms, a random solution space is used for task scheduling. With a few modifications, these algorithms can solve various optimization problems [21]. Meta-heuristic algorithms are problem-independent [22], and using them for fog task scheduling has shown promise in improving resource utilization and reducing latency in distributed fog computing environments. This section reviews several studies on meta-heuristic task scheduling in the fog environment.

Xu, al [23] proposed a task scheduling approach based on Laxity-Based Priority and Ant Colony System (LBP-ACS) in a cloud-fog environment. The LBP algorithm is applied to determine the priority of tasks, and the Constrained Optimization Algorithm based on the ACS (COA-ACS) is used for task scheduling. Simulation results showed that the proposed algorithm outperforms Greedy for Energy (GfE), Heterogeneous Earliest Finish Time (HEFT), and hybrid ant colony optimization with differential evolution algorithms in terms of reducing energy consumption and failure rate in scheduling

dependent tasks with mixed deadlines. However, the study only considered associated tasks, not independent ones.

Ghobaei-Arani, al [24] introduced a Moth-Flame Optimization (TS-MFO) algorithm for scheduling tasks in the fog environment. The main goal of the TS-MFO algorithm is to meet the QoS requirements of Cyber-Physical System (CPS) applications.

The algorithm identifies the best solutions to locate fog nodes. Using the iFogSim simulator, TS-MFO is compared with Particle Swarm Optimization (PSO), Non-dominated Sorting Genetic Algorithm-II (NSGA-II), and Bee Life Algorithm (BLA). Results revealed that the TS-MFO algorithm reduces the total execution time of tasks. However, the study does not consider energy consumption and communication cost.

Rafique, al [25] proposed a Novel Bio-Inspired Hybrid Algorithm (NBIHA) for efficient task scheduling and resource management in the fog environment. NBIHA combines Modified Particle Swarm Optimization (MPSO) and Modified Cat Swarm Optimization (MCSO). The algorithm discovers the best match of fog devices for an input task based on the requested memory and CPU time. Compared to First Come First Serve (FCFS), Shortest Job First (SJF), and MPSO, the NBIHA algorithm provides better results in terms of execution time, energy consumption, and average response time. However, this study does not address communication cost.

Hosseinioun, al [26] proposed an efficient hybrid algorithm based on invasive weed optimization and culture algorithms. The goal of this algorithm is to minimize energy consumption using Dynamic Voltage and Frequency Scaling (DVFS). The results confirmed that the proposed DVFS algorithm meets real-time requirements in heterogeneous systems.

Table 3: Summary Of The Task Scheduling In Metaheuristic-Based Mechanisms

Journal	Technique	Advantage	Weakness
Arisdakessian, et al. [1]	Intelligent scheduling based on game theory	Low energy consumption, Low execution time	High resource utilization
Xu, et al. [2]	Laxity-based priority and Ant Colony System	Low energy consumption	High execution time
Ghobaei-Arani, et al. [3]	Moth-Flame Optimization algorithm	Low execution time	High energy consumption, High cost
Rafique, et al. [4]	Bio-inspired hybrid algorithm	Low execution time, Low response time, Low energy consumption	High cost
Hosseinioun, et al. [5]	Invasive Weed Optimization and Cultural Algorithm	Low energy consumption, Low execution time	

2.4 An Overview of Deterministic Mechanisms

Deterministic mechanisms always produce the same output for a given input. Deterministic mechanisms for fog task scheduling involve using a predefined set of rules and algorithms to allocate tasks to fog nodes. An example of a deterministic approach is exhaustive search, where the entire search space is enumerated to find the optimal plan based on a given cost model. These mechanisms do not involve any randomness or probabilistic methods. While deterministic mechanisms are simple and easy to implement, they may not always result in optimal resource utilization or latency reduction.

This section discusses deterministic mechanisms for fog task scheduling. Kyung [27] presented a prioritized task distribution model considering static

and opportunistic fog nodes with respect to their mobility. In this model, delay-sensitive tasks are processed in static fog nodes, while delay-insensitive tasks are handled in opportunistic fog nodes. The results showed that the proposed model performs better than traditional models in terms of service response delay and outage probability. However, ranking fog nodes may not always accurately reflect their actual computing capabilities or workload, and prioritizing tasks could lead to lower-priority tasks being delayed or neglected, impacting overall system performance.

Tsai, al [28] proposed a linear transformation model to solve the nonlinear task assignment problem in cloud-fog systems. Various criteria, including execution time and operating costs, are considered for optimal task allocation. The model finds the optimal solution based on task requirements, nodes' processing speed, and nodes' resource usage cost. Although effective, the computational complexity of this approach grows rapidly as the problem size increases.

Caminero and Muñoz-Mansilla [29] introduced a network-aware scheduling algorithm to select the appropriate fog node to execute an application within a given deadline. The algorithm, an extension of the Kubernetes default scheduler, considers the network status in its scheduling decisions. Results showed that the algorithm executes all submitted tasks within the deadline, reaching an optimal solution. Similarly, Yin et al. [30] presented container-based task scheduling algorithms in fog computing, which improved resource utilization by minimizing delay.

Razaque, al [30] proposed an efficient hybrid algorithm to decrease energy consumption for the Mobile Fog-Based Cloud (MFBC). The algorithm uses a voltage scaling factor to reduce energy consumption and secures the identity of mobile cloud users using blockchain technology. The approach includes job-scheduling and machine power calculation modules to allocate tasks for mobile fog cloud users in a time-series fashion, improving throughput and latency.
It also proposes a secure key management scheme to ensure secure communication between fog nodes and the cloud. The performance of this algorithm is significantly better than state-of-the-art algorithms in terms of security, energy efficiency, throughput, and latency. However, the complexity of the architecture may require significant resources to implement.

Table 4: Summary Of The Task Scheduling In Deterministic Mechanisms.

Journal	Technique	Advantage	Weakness
Kyung [1]	Priority-based Task Distribution Model	Low response time, Low uncertainty	High complexity
Tsai, et al. [2]	Linear Transformation Model	Low latency	High resource usage, High execution time
Caminero and Munoz-Mansilla [3]	Bio-inspired Hybrid Algorithm	Low execution time, Low latency	High complexity
Razaque, et al. [4]	Priority-based Task Distribution Model	Low energy consumption, Low execution time, Low latency, High security	High resource usage

3. COST-AWARE TASK SCHEDULING

3.1 Challenges of Cost-Based Planning:

Before presenting the CAG algorithm in detail, it is essential to identify the main challenges associated with scheduling in a distributed environment:
- Latency constraints: Some IoT applications require near-instantaneous processing, making it essential to take latency into account.
- Variable costs: fog and cloud resources present heterogeneous costs depending on their availability, load and energy consumption.
- Dynamic resource allocation: The heterogeneous, dynamic nature of devices requires flexible, scalable task allocation.

These challenges justify the use of a genetic algorithm to find an optimized solution.

3.2 Rationale for Choosing Genetic Algorithms:

Genetic algorithms (GA) are chosen for their ability to solve complex optimization problems. Compared with simpler approaches such as Round-Robin or Trade-Off algorithms, GA offers the following advantages:

- Efficient exploration of the search space: identifies a solution close to the global minimum.
- Adaptability: can dynamically adjust to load variations and latency fluctuations.
- Multi-criteria optimization: simultaneously minimizes costs and maximizes success rates.

This algorithm exploits these strengths to allocate tasks optimally, while minimizing costs and meeting deadlines.

3.3 Comparison with Other Approaches

To situate the effectiveness of CAG in relation to other scheduling methods, the following table summarizes the key features:

Table 5: Comparison Of Task Scheduling Approaches In Fog-Cloud Environments

Approach	*Complexity*	*Cost optimization*	*Latency management*	Adaptability
Round-Robin [27]	Linear Complexity(O(n))	Low	Medium	Low
Trade-Off [31]	Log-Linear Complexity (O(n log n))	Average	Good	Average
CAG (proposed, this word)	Log-Linear Complexity (O(n log n))	High	Excellent	High

Explanation of Complexity Terms:

- **Round-Robin: Linear Complexity (O(n)):** For example, if you have 10 tasks, Round-Robin will take 10 units of time. If you have 100 tasks, it will take 100 units of time. This makes it less suitable for large-scale systems.
- **Log-Linear Complexity (O(n log n)):** For example, for 100 tasks, Trade-Off will take around 100 * log(100) time units, which is considerably better than Round-Robin for large sets of tasks.
- **Log-Linear Complexity (O(n log n)):** Like Trade-Off, CAG is effective for large task sets, but is smarter in the way it explores and exploits solutions, making it more efficient in terms of latency and cost management.

3.4: Cost-Aware Genetic Algorithm (CAG)

The cost-based genetic algorithm is a decision-making process used for task scheduling in distributed systems such as fog and cloud computing.

The goal is to optimize the allocation of resources to tasks over a defined period, considering multiple objectives, including processing cost and meeting deadlines for time-constrained tasks.

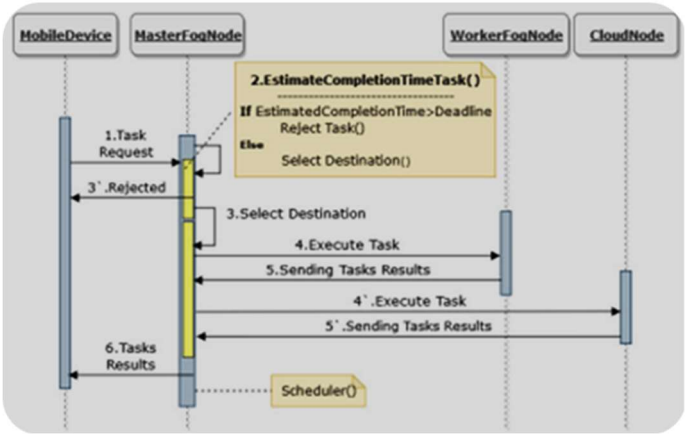

Figure 2: System Model Of Scheduling

The figure 2 illustrates the task scheduling process in a fog-cloud environment, managed by the MasterFogNode.

The scheduling system is managed by a central planning node called the MasterFogNode. When a task request is received, the MasterFogNode executes the scheduling algorithm.

This algorithm estimates the completion time of tasks based on the network and processing capabilities of available resources and the requirements of the tasks. If a task cannot be completed on time, it is rejected. Otherwise, it is sent

to the corresponding resources (fog nodes or cloud nodes) for processing. After processing, the results are returned to the MasterFogNode and then transmitted to the requester.

To provide an optimal solution, the MasterFogNode uses a cost-aware genetic algorithm (CAG) to assign a set of tasks $T=\{t_1,t_2,\ldots,t_n\}$ to fog and cloud nodes $D=\{d_1,d_2,\ldots,d_m\}$.

The algorithm takes an incoming list of tasks and available processing nodes as input. Each task has key characteristics such as arrival time, number of instructions, upload size, result data size, and deadline. The tasks are considered independent (Bag of Tasks - BoT) and can be scheduled independently of each other.

In the Genetic Algorithm, each possible solution is represented by a chromosome (individual). The representation of chromosomes is done through integer coding, which includes resource identifications. A set of chromosomes in a generation is called a population.

The CAG algorithm begins with defining the representation method (chromosome encoding). It randomly generates the initial generation of chromosomes (initial population). A fitness function is defined to measure the quality of solutions. This fitness function is used to determine which chromosomes will survive or be reproduced in subsequent generations. The search process, based on the fitness function, aims to provide the most suitable individuals.

The objective of the CAG algorithm is to reduce the task execution cost and increase the success rate of tasks completed within deadlines. To achieve this, the algorithm considers the total cost of computing resources and bandwidth used for task execution.

The first generation of chromosomes (initial population) is generated randomly. Another important step is defining the fitness function, which measures the quality of solutions by taking a candidate chromosome as input and producing an indication of how relevant the solution is relative to the problem's objectives. Thus, the fitness function determines which chromosomes will survive or be reproduced in subsequent generations, and the search process based on this fitness function attempts to provide the most suitable individuals.

To reduce the cost and increase the success rate, the fitness function is defined as follows:

- **Cost** représente le coût financier total de l'exécution des tâches.
- **SuccessRate** indique la fraction des tâches qui ont été accomplies avec succès.

$$f(x) = \frac{Cost}{SuccessRate} \quad (1)$$

The fitness function is designed to evaluate solutions based on these two criteria, aiming to minimize the cost while maximizing the task success rate. Chromosomes corresponding to low-cost solutions with a high success rate are more likely to be selected for reproduction, enabling the algorithm to converge towards more effective solutions over generations.

Figure 3: PSEUDO CODE OF CAG ALGORITHM

Algorithm 1 Pseudo Code of CAG Algorithm

Data: $T = \{T_1, T_2, ..., T_n\}$ and $D = \{d_1, d_2, ..., d_m\}$
Such that t_i is a task, and d_j is an available node.

Result: Mapping list between tasks and resources.

Function Scheduler (list of incoming tasks and available resources) is:
- Chromosomes encoding;
- Set Fitness Function;
- Population Size;
- Generate Initial Population;

Repeat
- Apply Genetic Operators;
- Evaluate Fitness Function f(x);

Until Termination Criteria

- Get Best Chromosomes;
- Mapping List = Best Chromosomes;

end

According to the CAG algorithm illustrated in Figure 3, the first step is to define the representation method (chromosome encoding). Each possible solution in the genetic algorithm is represented by a chromosome (individual). To represent the chromosomes, integer coding that includes resource identifications is used. A set of chromosomes in a generation is called a population.
The first generation of chromosomes (initial population) is generated randomly, another crucial step is defining the fitness function, which measures the quality of solutions by taking a candidate chromosome as input and determining how well that

solution aligns with the problem's objectives. Thus, the fitness function determines which chromosomes will survive or be reproduced in subsequent generations, and the search process based on this fitness function aims to provide the most suitable individuals.

To reduce cost and increase the success rate, we define the fitness function as follows:

- **Cost** represents the total financial expenditure dedicated to task execution.
- **SuccessRate** indicates the fraction of tasks that have been completed successfully.

Latency (T_L) and Processing Time (T_P) are calculated using Equation (2). In this equation, $I(t_i)$ represent the task size (number of instructions), and d_i denotes the index of the computing node. The waiting time (T_W) for each task to acquire the processor is computed using Equation (3). In this equation, t_i and t_j are executed on the same nodes, with t_j being executed before t_i. For simplicity, we assume that each node has a single processor.

$$T_p(t_i) = \frac{I(t_i)}{ProcessingRate(d_i)} + T_w(t_i) \quad (2)$$

$$T_w(t_i) = \sum_{j=1}^{K} T_p(t_j) \quad (3)$$

To calculate the latency (T_L) for task t_i, we must include the propagation time (PropagationTime) and the transmission time $T_N(t_i)$ as given in Equation (4). As commonly defined in communication systems, the propagation time is obtained by dividing the distance by the propagation speed. Additionally, the data transmission time $T_N(t_i)$ is calculated using Equation (5), where $S_i(t_i)$ and $S_o(t_i)$ represent the size of the input and output data for task ti , and Bd represents the bandwidth between the two points (i.e., master node and destination worker node). Thus, the data transmission time depends on the data size and bandwidth.

$$T_L(t_i) = T_N(t_i) + 2.PropagationTime(t_i) \quad (4)$$

$$T_N(t_i) = \frac{S_i(t_i)}{B_d} + \frac{S_o(t_i)}{B_d} \quad (5)$$

Incorporating energy consumption into the fitness function allows for a comprehensive evaluation of each solution.

Energy consumption is calculated based on the energy used by each fog and cloud node to process the tasks.

Thus, optimal solutions minimize latency, financial costs, and energy consumption, contributing to more sustainable and efficient solutions.

Equation (6) shows that the energy consumed by a task on a fog or cloud node is equal to the execution time $T_e(t_i)$ of that task multiplied by the energy consumption in idle mode of the fog or cloud device P_C:

$$E(t_i) = T_e(t_i) \cdot P_C \quad (6)$$

The processing cost is calculated in Equation (7), where C_P represents the processing cost per unit time on the destination node, C_B represents the cost of network bandwidth and C_E represents the cost associated with energy consumption.

$$\text{Cost} = \sum_{i=1}^{n} \left(T_p(t_i) \cdot C_p + T_L(t_i)\right) \cdot C_B + E(t_i) \cdot C_E \quad (7)$$

This equation accounts for the time spent processing tasks on the destination node, the network bandwidth used for data transmission, and the energy or resource costs involved in the execution of the tasks.

The proposed approach in this article is the Multi-Level IoT Tasks Scheduling (MLITS) model, which is an orchestration system for managing and allocating IoT tasks over the mist-fog-cloud architecture [31].

The MLITS model performs IoT tasks based on their deadline and the urgency of their execution, and it also performs various types of IoT tasks without rapidly consuming available sources while maintaining the resource usage balanced [32].

Additionally, the proposed scheduling model is compared with three other scheduling models, namely Min-Min, Credit-Based-Scheduling (CBS), and Earliest-Feasible-Deadline-First (EFDF), and the simulation results show that the MLITS model enhances the performance metrics, such as turnaround time, waiting time, and throughput [33].

Table 6: *Description Of Parameters Used In Scheduling Of Iot Tasks In The Fog.*

Parameter	Description
$I(t_i)$	*Represents the total number of instructions to be executed for task t_i.*

ProcessingRate(di)	*Represents the processing capacity of a given computing node* d_i
$T_w(t_i)$	*Calculated for a task ti, it is the time the task must wait before being executed on the same processor as other previously scheduled tasks.*
PropagationTime(t_i)	*Represents the propagation time for a bit to travel from the source to the destination.*
$T_M(t_i)$	*Represents the time required to transmit the input and output data of task* t_i *based on the data size* $S_i(t_i)$ *and* $S_o(t_i)$*, as well as the available bandwidth* B_d*.*
Cost	*Represents the total financial cost associated with the execution of tasks.*
$E(t_i)$	*Represents the total energy used to process a task.*
$T_L(t_i)$	*Measures the total delay experienced by a task between its sending and receiving.*

The cost-based genetic algorithm is a decision-making process used for task scheduling in distributed systems such as fog and cloud computing. The goal is to optimize the allocation of resources to tasks over a defined period, considering multiple objectives, including processing cost and meeting deadlines for time-constrained tasks.

4. SIMULATION SETTINGS

PureEdgeSim is a simulation framework that enables the study of Internet of Things environments at large scale, in a distributed, dynamic, and highly heterogeneous infrastructure. It also simulates the behavior of applications running on these systems. It provides realistic infrastructure models for research across the edge-to-cloud continuum, enabling evaluations of performance, cost, latency, and energy metrics. It encompasses all layers of edge computing modeling and simulation. It features a modular design in which each module addresses a particular aspect of the simulation.

For example, the Datacenters Manager module creates data centers, servers, and end devices with heterogeneity. The Location Manager handles geo-distribution and mobility. The Network Module manages bandwidth allocation and data transfer.
In this study, we configured the simulator to evaluate the performance of the proposed Cost-Aware Genetic (CAG) algorithm. We focused on metrics such as success rate, execution delay, cost, and energy consumption under various fog node configurations.

5. RESULTS AND DISCUSSIONS

To evaluate the performance of the proposed algorithm in case of increasing the number of fog nodes in simulation experiments, the number of cloud nodes are kept constant
(1 cloud nodes), while number of heterogeneous worker fog nodes varies between 200 and 600. The characteristics of the workload and nodes are kept constant in all simulations.
Also, to achieve a normal value, each of the experiments ran 3 times and the average values are provided.
Round-Robin (RR) is a simple but well-known algorithm that is one of the most common algorithms for resource allocation [34]. RR has been used by other papers in the literature for comparison [35], [36]. In addition to RR, Trade-Off, the algorithm divides the given circuit into subsystems such that each subsystem cannot exceed its reconfigurable capacity.
The union of the solutions of these subsystems is the solution of the whole system. Then, a greedy approach is proposed to find the solution for each subsystem [37], is used for comparison. In addition, to ensure system stability over time, simulations are repeated with different simulation duration.

5.1 Simulation Scenario

To evaluate the performance of the proposed algorithm in case of increasing the number of fog nodes
in simulation experiments, the number of cloud nodes are kept constant (1 cloud nodes), while number of
heterogeneous worker fog nodes varies between 200 and 600. The characteristics of the workload and nodes
are kept constant in all simulations. Also, to achieve a normal value, each of the experiments ran 3 times
and the average values are provided. Round-Robin (RR) is a simple but well-known algorithm that is one of

the most common algorithms for resource allocation[19]. RR has been used by other papers in the literature
for comparison[20], [21]. In addition to RR, Trade-Off, The algorithm divides the given circuit into subsys-
tems such that each subsystem cannot exceed its reconfigurable capacity. The union of the solutions of these
subsystems is the solution of the whole system. Then, a greedy approach is proposed to find the solution for
each subsystem [22], is used for comparison. In addition, to ensure system stability over time, simulations are
repeated with different simulation durations.

5.2 Evaluation Metrics

To evaluate the proposed approach, we consider the following metrics:

- **Success Rate**: The fraction of tasks that are completed within the deadline. This metric is calculated by dividing the number of successfully completed tasks by the total number of tasks.
- **Cost**: The financial cost incurred for the execution of tasks.

In this section, we investigate the performance of CAG algorithm in terms of Success Rate and Failed Rate and Average Execution Delay and Average Energy Consumption.

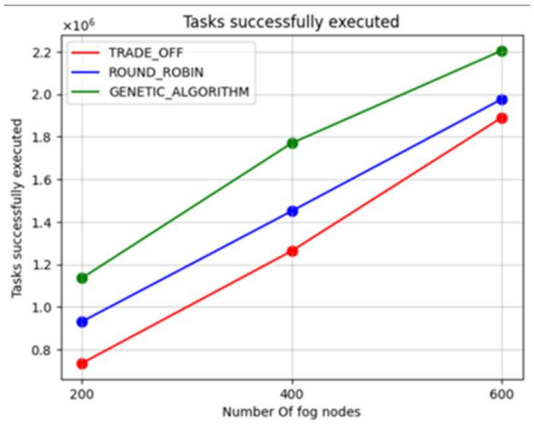

Figure 4: Tasks Successfully Executed

The figure 4 shows the number of successfully executed tasks as a function of the number of fog nodes for three algorithms: {ROUND_ROBIN}(red), {ROUND_ROBIN}(blue) and GENETIC ALGORITHM (green). It can be observed that the number of executed tasks increases with the number of fog nodes for all three algorithms.
GENETIC ALGORITHM significantly outperforms the other algorithms at every node level, demonstrating superior performance. {ROUND_ROBIN} ranks second, while {TRADE_OFF} is the least performant.

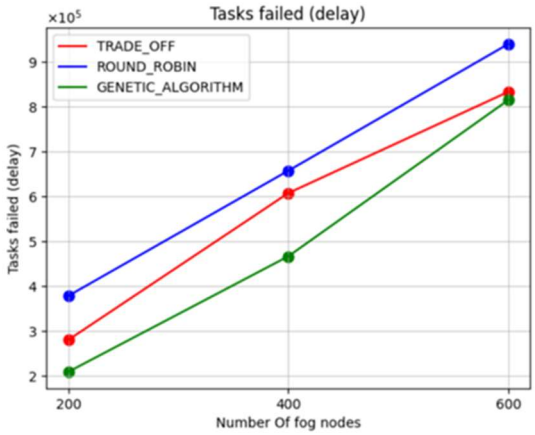

Figure 5: Tasks Failed (Delay)

The figure 5 presents the number of tasks that failed due to deadlines as a function of the number of fog nodes for three algorithms: {TRADE_OFF}(red), {ROUND_ROBIN}(blue), and {GENETIC_ALGORITHM}(green). It can be observed that the number of failed tasks increases with the number of fog nodes for all three algorithms. {GENETIC_ALGORITHM} shows the lowest number of failed tasks, indicating better deadline management. {ROUND_ROBIN} has the highest number of failed tasks, while {TRADE_OFF} falls in between the two.

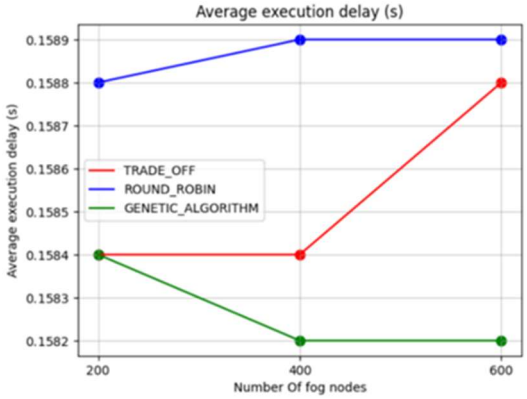

Figure 6: Average Execution Delay

The figure 6 shows the average execution delay for three algorithms ({TRADE_OFF}, {ROUND_ROBIN}, and {GENETIC_ALGORITHM}) based on the number of fog nodes. The {GENETIC_ALGORITHM} provides the best performance, with a slight decrease in delay as the number of nodes increases, stabilizing at 0.1582 seconds for 600 nodes. In comparison, the delay for the {ROUND_ROBIN} algorithm consistently increases, reaching 0.1589 seconds at 600 nodes. The {TRADE_OFF} algorithm shows a slight decrease in delay at 400 nodes but experiences a notable increase to 0.1589 seconds at 600 nodes. Thus, {GENETIC_ALGORITHM} is the most effective at minimizing execution delays.

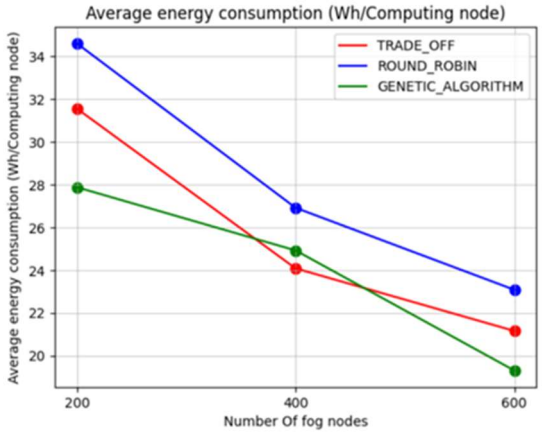

Figure 7: Average Energy Consumption

The figure 7 compares the average energy consumption per computing node based on the number of fog nodes for three scheduling algorithms: {TRADE_OFF}, {ROUND_ROBIN}, and {GENETIC_ALGORITHM}. Overall, energy consumption decreases as the number of fog nodes increases for all three algorithms.
However, the {GENETIC_ALGORITHM} stands out with significantly lower energy consumption compared to the T{RADE_OFF} and {ROUND_ROBIN} algorithms, indicating better energy efficiency. This observation highlights the advantage of the genetic algorithm in reducing energy consumption in fog-cloud environments.

6. CONCLUSION

The fog computing paradigm continues to reshape the execution models of distributed systems, especially in scenarios involving IoT and latency-sensitive services. Efficient task scheduling is a critical challenge in a cloud-fog ecosystem. In this paper, we addressed the problem of efficient task scheduling by proposing a genetic algorithm-based method called CAG, where fog nodes collaborate with rented cloud nodes to efficiently execute large-scale offloading applications. We have proposed a genetic-based algorithm called CAG, which balances the trade-off between throughput and cost for hard real-time applications.
To assess the performance of our proposed approach, we made modifications to the iFogSim simulator and evaluated CAG under various scenarios to examine its impact on performance. The results indicate that CAG outperforms the Round-Robin (RR) and Minimum Response Time algorithms in terms of financial cost and success rate.

For future work, we aim to extend this approach by exploring energy-aware enhancements and considering task dependencies (workflow scheduling). Additionally, we also plan to evaluate the model under realistic workloads, including mobility scenarios and dynamic resource availability.

REFERENCES:

[1] Somayya Madakam, R. Ramaswamy, and Siddharth Tripathi. Internet of Things (IoT): A Literature Review. Journal of Computer and Communications, 3(5):164–173, May 2015. doi: 10.4236/jcc.2015. 35021. URL https://www.scirp.org/journal/paperinformation? Paper id=56616.Number: 5 Publisher: Scientific Research Publishing.

[2] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Swami Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. ArXiv, abs/1207.0203,2012. URL https://api.semanticscholar.org/CorpusID:204982032.

[3] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pages 13–16, 2012.

[4] Flavio Bonomi, Rodolfo A. Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In MCC '12, 2012. URL https://api.semanticscholar.org/Corpus ID: 207196503.

[5] Ashkan Yousefpour, Ashish Patil, Genya Ishigaki, Inwoong Kim, Xi Wang, Hakki C Cankaya, Qiong Zhang, Weisheng Xie, and Jason P Jue. Fogplan: A lightweight qos-aware

dynamic fog service provisioning framework. IEEE Internet of Things Journal, 6(3):5080–5096, 2019.

[6] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. Software: Practice and Experience, 47(9):1275–1296, 2017.

[7] Amir Masoud Rahmani, Elham Azhir, Saqib Ali, Mokhtar Mohammadi, Omed Hassan Ahmed, Marwan Yassin Ghafour, Sarkar Hasan Ahmed, and Mehdi Hosseinzadeh. Artificial intelligence approaches and mechanisms for big data analytics: a systematic study. PeerJ Computer Science, 7: e488, April 2021. ISSN 2376-5992. doi: 10.7717/peerj-cs.488. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8053021/.

[8] Amir Masoud Rahmani, Elham Azhir, Morteza Naserbakht, Mokhtar Mohammadi, Adil Hussein Mohammed Aldalwie, Mohammed Kamal Majeed, Sarkhel H. Taher Karim, and Mehdi Hosseinzadeh. Correction to: Automatic COVID-19 detection mechanisms and approaches from medical images: a systematic review. Multimedia Tools and Applications, 81(20):28799–28800, 2022. ISSN 1380-7501. doi: 10.1007/s11042-022-13374-1.

[9] Harwant Singh Arri, Ramandeep Singh, Sudan Jha, Deepak Prashar, Gyanendra Prasad Joshi, and Ill Chul Doo. Optimized task group aggregation-based overflow handling on fog computing environment using neural computing. Mathematics, 9(19):2522, 2021.

[10] He Li, Kaoru Ota, and Mianxiong Dong. Deep reinforcement scheduling for mobile crowdsensing in fog computing. ACM Transactions On Internet Technology (TOIT), 19(2):1–18, 2019.

[11] Pegah Gazori, Dadmehr Rahbari, and Mohsen Nickray. Saving time and cost on the scheduling of fog-based iot applications using deep reinforcement learning approach. Future Generation Computer Systems, 110:1098–1115, 2020.

[12] Shashank Swarup, Elhadi M Shakshuki, and Ansar Yasar. Energy efficient task scheduling in fog environment using deep reinforcement learning approach. Procedia Computer Science, 191:65–75, 2021.

[13] Elhaou, H., Oukissou, Y., Ait Omar, D. *et al.* Machine learning for user mobility management in a mobile fog computing environment. *Cluster Comput* **28**, 305 (2025). https://doi.org/10.1007/s10586-024-05076-0

[14] Bushra Jamil, Humaira Ijaz, Mohammad Shojafar, and Kashif Munir. Irats: A drl-based intelligent priority and deadline-aware online resource allocation and task scheduling algorithm in a vehicular fog network. Ad hoc networks, 141:103090, 2023.

[15] Kai Li, Wei Ni, Xin Yuan, Alam Noor, and Abbas Jamalipour. Deep-graph-based reinforcement learning for joint cruise control and task offloading for aerial edge internet of things (EDGEIOT). IEEE Internet of Things Journal, 9(21):21676–21686, 2022.

[16] Jean-François Cordeau, Michel Gendreau, and Gilbert Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks: An International Journal, 30(2):105–119, 1997.

[17] Sarhad Arisdakessian, Omar Abdel Wahab, Azzam Mourad, Hadi Otrok, and Nadjia Kara. Fogmatch: An intelligent multi-criteria iot-fog scheduling approach using game theory. IEEE/ACM Transactions on Networking, 28(4):1779–1789, 2020.

[18] Sadoon Azizi, Mohammad Shojafar, Jemal Abawajy, and Rajkumar Buyya. Deadline-aware and energy-efficient iot task scheduling in fog computing systems: A semi-greedy approach. Journal of network and computer applications, 201:103333, 2022.

[19] Entesar Hosseini, Mohsen Nickray, and Shamsollah Ghanbari. Optimized task scheduling for cost-latency trade-off in mobile fog computing using fuzzy analytical hierarchy process. Computer Networks, 206: 108752, 2022.

[20] Fulong Xu, Zhenyu Yin, Ai Gu, Yue Li, Haoyu Yu, and Feiqing Zhang. Adaptive scheduling strategy of fog computing tasks with different priority for intelligent production lines. Procedia Computer Science, 183:311–317, 2021.

[21] Elham Azhir, Nima Jafari Navimipour, Mehdi Hosseinzadeh, Arash Sharifi, Mehmet Unal, and Aso Darwesh. Join queries optimization in the distributed databases using a hybrid multi-objective algorithm. Cluster Computing, pages 1–16, 2022.

[22] Fatos Xhafa and Ajith Abraham. Computational models and heuristic methods for grid

scheduling problems. Future generation computer systems, 26(4):608–621, 2010.

[23] Jiuyun Xu, Zhuangyuan Hao, Ruru Zhang, and Xiaoting Sun. A method based on the combination of laxity and ant colony system for cloud-fog task scheduling. IEEE access, 7:116218–116226, 2019.

[24] Mostafa Ghobaei-Arani, Alireza Souri, Fatemeh Safara, and Monire Norouzi. An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing. Transactions on Emerging Telecommunications Technologies, 31(2): e3770, 2020.

[25] Hina Rafique, Munam Ali Shah, Saif Ul Islam, Tahir Maqsood, Suleman Khan, and Carsten Maple. A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing. IEEE access, 7:115760–115773, 2019.

[26] Pejman Hosseinioun, Maryam Kheirabadi, Seyed Reza Kamel Tabbakh, and Reza Ghaemi. A new energy aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm. Journal of Parallel and Distributed Computing, 143:88–96, 2020.

[27] Yeunwoong Kyung. Prioritized task distribution considering opportunistic fog computing nodes. Sensors, 21(8):2635, 2021.

[28] Jung-Fa Tsai, Chun-Hua Huang, and Ming-Hua Lin. An optimal task assignment strategy in cloud-fog computing environment. Applied Sciences, 11(4):1909, 2021.

[29] Agustın C Caminero and Rocıo Munoz-Mansilla. Quality of service provision in fog computing: Network-aware scheduling of containers. Sensors, 21(12):3978, 2021.

[30] Abdul Razaque, Yaser Jararweh, Bandar Alotaibi, Munif Alotaibi, Salim Hariri, and Muder Almiani. Energy-efficient and secure mobile fog-based cloud for the internet of things. Future Generation Computer Systems, 127:1–13, 2022.

[31] Luxiu Yin, Juan Luo, and Haibo Luo. Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing. IEEE Transactions on Industrial Informatics, 14(10):4712–4721, 2018.

[32] Qian Ren, Kui Liu, and Lianming Zhang. Multi-objective optimization for task offloading based on network calculus in fog environments. Digital Communications and Networks, 8(5):825–833, 2022.

[33] Brahim Syed. Hamm: A hybrid algorithm of min-min and max-min task scheduling algorithms in cloud computing. International Journal of Recent Technology and Engineering (IJRTE), 9:209–218, 2020.

[34] William Stallings. Operating Systems: Internals and Design Principles, 9/e. Pearson IT Certification, 2018.

[35] Guoqi Xie, Xiongren Xiao, Hao Peng, Renfa Li, and Keqin Li. A survey of low-energy parallel scheduling algorithms. IEEE Transactions on Sustainable Computing, 7(1): 27–46, 2021.

[36] Muhammad Usman Sana and Zhanli Li. Efficiency aware scheduling techniques in cloud computing: a descriptive literature review. PeerJ Computer Science, 7: e509, 2021.

[37] Shaojun Wei, Xinhan Lin, Fengbin Tu, Yang Wang, Leibo Liu, and Shouyi Yin. Reconfigurability, why it matters in ai tasks processing: A survey of reconfigurable ai chips. IEEE Transactions on Circuits and Systems I: Regular Papers, 70(3):1228–1241, 2022.