Optimizing Task Scheduling in Fog Computing with Deadline Awareness

Mohammad Sadegh Sirjani
Department of Computer Science
University of Texas at San Antonio
San Antonio, USA
mohammadsadegh.sirjani@utsa.edu

Somayeh Sobati-Moghadam
Department of Computer Engineering
Hakim Sabzevari University
Sabzevar, Iran
s.sobati@hsu.ac.ir

***Abstract*—The rise of Internet of Things (IoT) devices has led to the development of numerous applications that require quick responses and low latency. Fog computing has emerged as a solution for processing these IoT applications, but it faces challenges such as resource allocation and job scheduling. Therefore, it is crucial to determine how to assign and schedule tasks on Fog nodes. A well-designed job scheduling algorithm can help decrease energy usage and improve response times for application requests. This work aims to schedule tasks in IoT while minimizing the total energy consumption of nodes and enhancing the Quality of Service (QoS) requirements of IoT tasks, taking into account task deadlines. Initially, this paper classifies the Fog nodes into two categories based on their traffic level: low and high. It schedules low-deadline tasks on low-traffic-level nodes using an Improved Golden Eagle Optimization (IGEO) algorithm, an enhancement of the Golden Eagle Optimization Algorithm that utilizes genetic operators for discretization. High-deadline tasks are processed on high-traffic nodes using reinforcement learning (RL). This combined approach is called the Reinforcement Improved Golden Eagle Optimization (RIGEO) algorithm. Experimental results demonstrate that the proposed algorithms optimize system response time, total deadline violation time, and resource and system energy consumption compared to other state-of-the-art algorithms.**

***Index Terms*—Internet of Things, Fog Computing, Job Scheduling, Golden Eagle Optimization Algorithm, Reinforcement Learning**

I. Introduction

In recent years, the Internet of Things (IoT) has become a prominent technology in the internet and networking industry [1]. It consists of a system of linked physical devices, like household items and cars, that have sensors, software, and online connectivity [2]. Although IoT has been expanding quickly, it is encountering obstacles such as traffic congestion, delays, and excessive energy usage. In the realm of IoT, a deadline refers to a specific time by which a task or operation must be completed. Adhering to deadlines is crucial in real-time applications where prompt responses are vital, such as e-health in the healthcare sector, smart grid in energy management, livestock monitoring in agriculture, and smart traffic in transportation management. These time-sensitive IoT applications typically have stringent quality of service (QoS) requirements and necessitate processing capabilities that surpass the limitations of IoT devices, which often have significant resource constraints [3]. Effectively managing resources in IoT and minimizing energy usage while meeting task deadlines can be a challenging task [4]. Fog computing has emerged as a solution to address challenges in IoT networks by providing computational capabilities at the edge, reducing delays and energy consumption [5], [6]. However, using fog presents its own set of challenges, as resources are diverse and limited, necessitating efficient resource scheduling and allocation strategies. Developing practical approaches for allocating resources in fog nodes is essential for optimal performance [7]. However, despite these advancements, many existing solutions often rely on a single optimization strategy or fail to fully account for the dynamic and heterogeneous nature of real-world IoT-Fog environments, particularly in terms of varying network traffic levels and diverse task deadline requirements [8]–[11]. To address these limitations, this work proposes a novel adaptive task scheduling framework, the Reinforcement Improved Golden Eagle Optimization (RIGEO) algorithm. A key innovation of RIGEO lies in its dynamic classification of Fog Nodes (FNs) into low-traffic and high-traffic categories based on real-time network conditions. Low-traffic nodes are characterized by average network traffic falling below a predefined threshold, while high-traffic nodes are those that exceed this average. This allows for a tailored scheduling approach, leveraging the strengths of different optimization paradigms for distinct operational scenarios. Specifically, for low-deadline tasks on low-traffic nodes, RIGEO employs an enhanced meta-heuristic algorithm, the Improved Golden Eagle Optimization (IGEO), which is designed for rapid and efficient resource allocation to meet stringent time constraints. Conversely, for high-deadline tasks destined for high-traffic nodes, Reinforcement Learning (RL) is utilized, offering robust decision-making capabilities to navigate complex and congested environments while still adhering to deadlines. The proposed RIGEO algorithm operates within the following three-layered architecture: cloud at the top, fog in the middle, and IoT devices at the bottom. IoT devices enable a wide range of devices, including wearable technology, RFID tags, sensors in automobiles and security systems, smart home appliances, thermostats, meters, and industrial equipment. The fog environment consists of a fog controller (FC), also known as a broker, and a network for managing resources and scheduling tasks. The cloud environment includes virtual machines with significant computing and storage capabilities. This architecture is illustrated in Figure 1.

Within this architectural context, since task scheduling has been introduced as an NP-hard problem, conventional methods are unable to solve it [12], [13]. Therefore, it is suggested to utilize machine learning and heuristic methods to solve this problem [14].

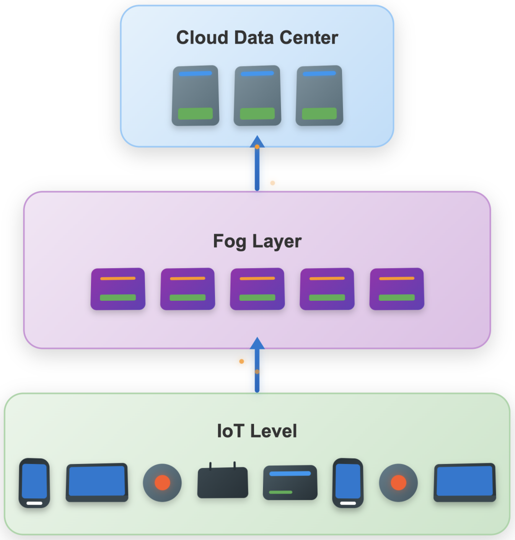

Fig. 1: The architecture of the IoT-Fog-Cloud network.

There are multiple challenges in the field of task scheduling, including cost optimization, energy efficiency, meeting deadline constraints, reducing latency, ensuring reliability, and achieving fault tolerance. Many studies have been conducted in each of these areas, and some of them will be mentioned in this section.

A. *Energy Efficiency*

Authors in [15] proposed a new task scheduling algorithm for fog computing environments, called MEETS. The algorithm employs a two-step approach: first, it determines the optimal task allocation to fog nodes, and then it schedules the tasks on each node accordingly. The results show that MEETS can significantly reduce energy consumption while still meeting all task deadlines. However, they considered homogeneous fog networks and assumed that the task arrival rate is constant, which may not be realistic in practice.

The authors in [16] proposed an algorithm named Energy-Efficient Task Scheduling in Fog Computing based on Particle Swarm Optimization (EETSPSO). The results demonstrated that EETSPSO outperforms existing approaches in terms of makespan, energy consumption, and execution time. However, as fog nodes are resource-constrained, the algorithm's performance may vary across different real-world scenarios with varying resource limitations.

B. *Deadline constrains*

The task scheduling approach proposed in [7] incorporates two semi-greedy algorithms: Priority-aware Semi-Greedy (PSG) and PSG with Multi-start Procedure (PSG-M). These algorithms aim to optimize energy consumption and minimize deadline violations for IoT tasks by effectively mapping them to fog nodes while ensuring compliance with Quality of Service (QoS) requirements. Notably, the PSG-M algorithm demonstrates a significant reduction in deadline violation time. However, the performance of these algorithms may vary under different real-world scenarios characterized by diverse resource constraints.

In [17], Louail et al. introduced a dynamic task scheduling algorithm that considered both deadline constraints and task frequency to improve the efficiency of fog nodes in smart factories. By considering task frequency, the algorithm optimizes the utilization of available resources, thereby reducing idle time and enhancing overall efficiency. However, the algorithm was complex to implement and required significant computational resources, which could be a challenge in resource-constrained environments.

C. *Latency Reduction*

Authors in [18] proposed an Energy-Efficient Internet of Medical Things to Fog Interoperability of Task Scheduling (EEIoMT) framework. This framework aims to schedule tasks efficiently by ensuring that critical tasks are executed within their deadlines while balancing energy consumption for other tasks. The framework utilizes Electrocardiogram (ECG) sensors to monitor heart health in a smart city environment. By processing data closer to the end user, the framework reduces latency and improves response times for IoMT applications. However, the study focuses on specific scenarios and algorithms, and other factors may also affect the framework's performance in different real-world scenarios.

The task scheduling approach proposed in [19] presents a scheduling algorithm called the Priority Queue Fuzzy Analytical Hierarchy Process (PQFAHP). This algorithm uses fuzzy logic and the Analytical Hierarchy Process (AHP) to prioritize tasks based on multiple criteria, including completion time, energy consumption, RAM usage, and deadlines. The goal was to achieve an optimal trade-off between cost and latency, ensuring efficient task execution in mobile fog computing environments. However, the algorithm may be complex to implement and require significant computational resources, which could be a challenge in resource-constrained environments.

D. *Cost optimization*

The authors in [8] proposed a Multi-Objective Cost-Aware Discrete Grey Wolf Optimization-based Algorithm (MoDGWA) to tackle the task scheduling challenges in IoT applications. This algorithm focuses on optimizing execution time, costs, and reliability by incorporating both execution cost and monetary cost models. Despite its effectiveness, the algorithm's complexity and high computational demands could pose significant challenges in environments with limited resources.

The authors in [20] introduced a Cost-Aware Genetic-Based (CAG) task scheduling algorithm tailored for fog-cloud environments. This algorithm is designed to improve cost efficiency in real-time applications with stringent deadlines by optimizing the allocation of services and resources within the three-tier IoT architecture. While it demonstrates effectiveness in handling delay-sensitive applications, the study is limited to specific scenarios and metrics, leaving room for potential variability in the algorithm's performance under different real-world conditions.

E. Reliability AND fault tolerant

In [9], the authors propose a Dynamic Fault Tolerant Learning Automata (DFTLA) task scheduling approach. This approach uses variable-structure learning automata to efficiently assign tasks to fog nodes, ensuring reliable execution while optimizing response time and energy consumption. The performance of the DFTLA scheduler is evaluated and compared with three baseline methods, demonstrating its effectiveness in ensuring reliable task execution and optimizing performance. However, the authors in [10] proposed the Load Balanced Service Scheduling Approach (LBSSA), designed to efficiently allocate customer requests to resources in cloud-fog environments. This approach categorizes requests into three groups: real-time, important, and time-tolerant, and incorporates resource failure rates to enhance reliability. However, the study primarily focuses on specific scenarios and metrics, potentially overlooking factors that might impact the mechanism's performance in varied real-world situations. An overview of these works is presented in Table I.

This article optimizes system response time, overall deadline violation time, and energy consumption in Fog Nodes (FNs). FNs are classified as low-traffic or high-traffic based on the distribution of IoT devices and user mobility. Tasks are labeled with low or high deadlines; low-traffic FNs process low-deadline tasks, while high-traffic FNs process high-deadline tasks. Reinforcement Learning (RL) is utilized for scheduling high-deadline tasks. For low-deadline tasks, the high-speed Golden Eagle Optimization (GEO) meta-heuristic algorithm is employed, and it is further refined into Improved Golden Eagle Optimization (IGEO) using genetic operators for discrete task scheduling. These algorithms form the core of the proposed RIGEO approach. Findings demonstrate reduced energy consumption, enhanced Quality of Service (QoS), meeting task deadlines, and minimized response times.

The results show that this approach reduces energy consumption, enhances Quality of Service by meeting task deadlines, and reduces response time.

In summary, the study involved categorizing FNs by traffic, introducing IGEO for low-traffic task scheduling, utilizing RL for high-traffic task scheduling, and enhancing energy efficiency while fulfilling task deadlines and reducing response times in the IoT-fog network. Findings demonstrate reduced energy consumption, improved Quality of Service (QoS) through deadline adherence, and minimized response time. The remainder of this article details the methodology and modeling in Section II, presents optimization results in Section III, and concludes in Section IV.

Ref.	**Year**	**Factors**				
		Energy Eff.	**Deadline Const.**	**Latency Red.**	**Cost Opt.**	**Reliability**
[15]	2018	✓	✓	×	×	×
[16]	2023	✓	×	×	×	×
[7]	2022	✓	✓	×	×	✓
[17]	2020	×	✓	×	×	×
[18]	2022	✓	✓	✓	×	×
[19]	2022	✓	✓	✓	×	×
[8]	2024	×	×	×	✓	✓
[20]	2020	×	✓	×	✓	×
[11]	2017	×	×	×	✓	×
[9]	2020	×	×	×	×	✓
[10]	2021	×	×	×	×	✓
Ours		✓	✓	✓	×	×

TABLE I: Comparison of factors considered in related work

II. Methodology

This section outlines the Task Scheduling problem in mathematical terms and provides a detailed description of the method used.

A. Problem Formulation

Given a set of n independent tasks $T = \{t_1, t_2, \ldots, t_n\}$ that need to be allocated to m heterogeneous FNs $F = \{f_1, f_2, \ldots, f_m\}$, the goal is to assign tasks to nodes in a way that minimizes energy consumption and ensures that task deadlines are met. The fog network is represented as a graph $G = (V, E)$ with connection links between nodes. Each task t_i has a deadline d_i in milliseconds. The problem is to find an optimal mapping $M : T \rightarrow F$ to achieve these objectives. The following describes how to obtain some of the parameters used in the article.

1) Response time: The response time for each task t_i is determined by the sum of the propagation delay, transmission delay, execution time, and queue waiting time as shown in Eq. (1) [2].

$$R_i = P_i + T_i + E_i + Q_i \tag{1}$$

Propagation delay, represented by P_i, is the duration it takes for a signal or data to move from one location to another within a communication channel. The transmission delay, T_i, is the time needed to transmit all bits of the task, while E_i indicates the processing time in the CPU, which significantly impacts a task's response time. Q_i signifies the amount of time a task spends in a queue after being assigned to a FN, waiting for higher-priority tasks assigned to the FN to be completed.

2) Deadline violation: When assessing the efficiency of time scheduling, it is essential to consider how many IoT tasks meet their deadlines compared to the total amount of time that deadlines are missed for a group of tasks sent to a broker within a specific time period [21]. The deadline violation time for each task can be calculated using Eq. (2), and the total deadline violation time for all functions can be determined using Eq. (3).

$$DV_i = \max(0, R_i - d_i) \tag{2}$$

$$DV_{total} = \sum_{i=1}^{n} DV_i \tag{3}$$

3) Energy Consumption: In this context, we are focusing on the amount of energy used by FNs to complete all tasks. This includes calculating the energy spent by an FN by adding up the energy consumed during active (E_{active}) and idle modes (E_{idle}). α and β are coefficients that the user determines. Eq. (4) shows the energy consumption by f_j and Eq. (5) shows the energy consumption for all the system [21].

$$E_j = \alpha \cdot E_{active,j} + \beta \cdot E_{idle,j} \tag{4}$$

$$E_{total} = \sum_{j=1}^{m} E_j \tag{5}$$

B. Proposed Method (RIGEO)

To maintain efficient resource allocation and effectively establish rules for Quality of Service (QoS) in the IoT and fog network, it is crucial to monitor and analyze network traffic. Focusing on network traffic becomes increasingly important as the number of Internet applications and the complexity of traffic grow. By understanding and managing network traffic, network quality can be enhanced. In this work, we assume that the network is divided into two classes: low- and high-traffic. First, a simple method is used to classify FNs based on their traffic. The average network traffic between FNs is calculated, and if the network traffic is below the average, it is classified as low traffic; otherwise, it is classified as high traffic. This average traffic level is periodically monitored and updated to reflect current network conditions dynamically. While this periodic re-evaluation allows adaptation to evolving traffic patterns and potential node reclassification during longer execution, the real-time handling of sudden, unpredictable traffic spikes and their impact on ongoing tasks remains a complex challenge for future investigation. It should be considered that when network traffic is high, faster algorithms should be used to ensure that tasks are executed within their deadlines.

The proposed method in this work considers deadline tasks in two stages and is referred to as the Reinforcement Learning Improved Golden Eagle Optimization (RIGEO) algorithm.

- It executes algorithms with a low deadline time on FCs with lower traffic to prevent tasks from waiting in the queue.
- To expedite the execution of algorithms with shorter deadline times, meta-heuristic algorithms are utilized. However, as heuristic algorithms introduce significant overhead on the system, there is no justification for tolerating this overhead for tasks with more available deadline time. Therefore, the Reinforcement Learning algorithm is implemented.

1) Reinforcement Learning: Reinforcement Learning (RL) is one of the main types of Machine Learning, which includes Supervised, Unsupervised, and Semi-supervised Learning [22]. RL algorithms aim to train an agent to make optimal decisions in a complex environment by providing feedback and observing the outcomes of its actions. In this model, the agent interacts with the environment, receives input in the form of a reinforcement signal, and learns to maximize long-term rewards by selecting actions that lead to the best outcomes. By learning from experience and utilizing various algorithms, the agent enhances its decision-making capabilities over time. The model comprises a finite set of environment states, agent actions, and reinforcement signals, along with an input function that determines how the agent perceives the environment state.

Initially, we consider an array with a length equal to the number of tasks with high deadlines. Each cell of the array refers to a task and includes an FC that should be able to accommodate that task. Each FC can accommodate one or more tasks. Initially, all elements of the array are randomly determined. This array changes over time in response to environmental feedback. The variety is given to RL as a solution, and the algorithm evaluates it. If the result obtained from state $T+1$ is better than the result obtained from state T, a reward is returned to the system; otherwise, a penalty is returned to the system. Based on the feedback received from the environment, the best scheduling is ultimately determined.

2) Improved Golden Eagle Optimization Algorithm: Meta-heuristic algorithms are renowned for their speed and ability to quickly find satisfactory solutions. In this work, a meta-heuristic algorithm called Golden Eagle Optimization (GEO) is used for scheduling low-deadline tasks by low-traffic FCs. GEO is a swarm-based meta-heuristic method inspired by the hunting behavior of golden eagles. We specifically chose the Golden Eagle Optimization (GEO) algorithm as the foundation for IGEO due to its demonstrated effectiveness in global optimization problems and its inherent characteristics that align well with the requirements of scheduling low-deadline tasks on low-traffic Fog Nodes. Meta-heuristic algorithms like GEO are highly valued for their speed and ability to find satisfactory solutions quickly, which is crucial for tasks with strict and short deadlines, where processing speed is critical to avoid violations. Furthermore, GEO's unique search mechanism, inspired by hunting behavior, allows for an effective balance between exploration (searching new areas) and exploitation (refining existing solutions), a balance further enhanced in IGEO through the integration of genetic operators for discrete task assignment [23]. It adjusts speed during different stages of hunting to efficiently find the global optimum and avoid local optima in solving global optimization problems.

To be effective, GEO must be discretized since task scheduling is a discrete problem. One method of discretizing a metaheuristic algorithm is by incorporating genetic operators. Using genetic operators in addition to the algorithm's discretization enhances its ability to explore and exploit, enabling it to find the optimal solution in the shortest possible time.

In the Golden Eagle Optimizer, each golden eagle selects a target to attack, calculates its cruising vector based on this target, and circles around the best location seen by its chosen prey. The prey represents the best solution found by the population of eagles, and each eagle keeps track of the best solution it has discovered. The attack vector for each eagle is determined by its current position and the location of its prey. The attack vector for golden eagle i is calculated through Eq. (6).

$$\vec{A_i} = \vec{X}_f^* - \vec{X}_i \tag{6}$$

The step vector for golden eagle i at iteration t is given by Eq. (7), where C is the destination point and C_k represents the k-th element of C. Where $\vec{r}_1$ and $\vec{r}_2$ are random vectors, whose elements are in the interval $[0, 1]$, p_a and p_c are the attack coefficient and cruise coefficient, respectively; p_a and p_c are user-defined. $\|\vec{A}_i\|$ is the size of the attack vector and $\|\vec{C}_i\|$ is the size of the cruise vector. The position of the golden eagles in iteration $t+1$ is calculated simply by adding the step vector in iteration t to the positions in iteration t. Eq. (7) shows the step vector for the golden eagle.

$$\Delta x_i = \vec{r}_1 p_a \frac{\vec{A}_i}{\|\vec{A}_i\|} + \vec{r}_2 p_c \frac{\vec{C}_i}{\|\vec{C}_i\|} \tag{7}$$

$$x^{(t+1)} = x^t + \Delta x_i^t \tag{8}$$

Where $\frac{\vec{A}_i}{\|\vec{A}_i\|}$ and $\frac{\vec{C}_i}{\|\vec{C}_i\|}$ are unit vectors. They indicate the direction of the vector but don't provide any information about the size of the vector. Therefore, the size of the cruise and attack vectors depends only on $\vec{r}_1 p_a$ and $\vec{r}_2 p_c$. If $|r_1 p_a| < |r_2 p_c|$, the tendency to cruise in the algorithm is higher, so we use genetic operators such as mutation that have a high search power in the environment. If $|r_1 p_a| > |r_2 p_c|$, the tendency to attack is higher, so we use genetic operators like crossover that have a higher attack inclination.

When the step vector in Eq. (7) is negative, the GEO tends more towards exploration. When the step vector is positive, the algorithm tends towards exploitation. Therefore, in IGEO, we employ different genetic operators to enhance GEO by taking into account this property. In IGEO, when the step vector is negative, we use a mutation operation instead of an addition operation in Eq. (8), which has a greater tendency towards exploration. In mutation, parts of the parent's genetic material are randomly altered. Figure 2 shows the process of the mutation operator.

Eq. (9) shows the IGEO operation in the case of a negative step vector. Where x_{best} is the best location that has been found so far, while x_t is the location found in the current iteration.

$$x_{t+1} = \begin{cases} \text{Mutation}(x_{best}) & \text{if } r \geq 0.5 \\ \text{Mutation}(x_t) & \text{if } r < 0.5 \end{cases} \tag{9}$$

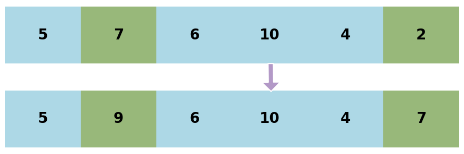

Fig. 2: Mutation operator

When the step vector is positive, the algorithm tends to be more inclined towards efficiency, so operators such as crossover, which are more efficient, are used. In crossover, genes from two parents are exchanged based on predetermined cutting points. Figure 3 shows two types of crossover. Eq. (10) shows how to update the new location in IGEO in the case of a positive operation.

$$x_{t+1} = \begin{cases} \text{single-point crossover}(x_{best}, x_t) & \text{if } r \geq 0.5 \\ \text{two-point crossover}(x_{best}, x_t) & \text{if } r < 0.5 \end{cases} \tag{10}$$

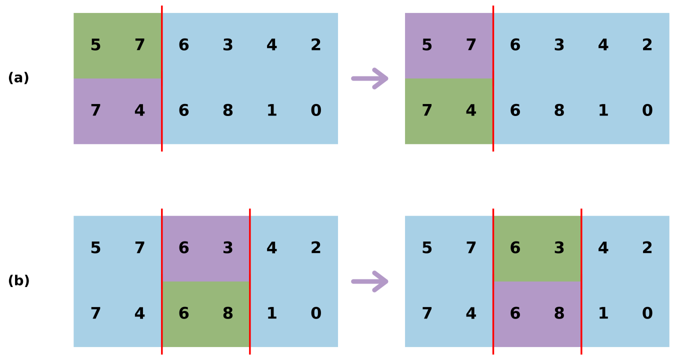

Fig. 3: Crossover operation. (a)One-point crossover. (b)Two-point crossover

In the algorithm 1, low-deadline tasks are sent to the low-traffic FCs and are processed using IGEO. High-deadline tasks are sent to high-traffic FCs and are processed using RL.

III. Results

This section is followed by an appraisal of the RIGEO approach's performance, as analyzed through various metrics. The experimental configuration utilizes a laptop equipped with an Intel Core i7 processor operating at 2.80 GHz, 16 GB of RAM, and the Windows 10 64-bit operating system. The algorithms were implemented as simulations within the MATLAB 2016 environment, leveraging its computational capabilities to model the fog computing scenario, and ran 50 times using parallel processors.

The results are compared against algorithms such as GEO [24], GWO [25], WCLA+GA [7], and ETFC [26] algorithms to showcase the performance of the RIGEO method.

This study conducted simulations in a fog environment using interconnected processing nodes with different energy

Algorithm 1 RIGEO Task Scheduling Algorithm

```
Require: Task $T$ with deadline $D$, Set of Fog Computing nodes $FC = \{fc_1, fc_2, \ldots, fc_n\}$
Ensure: Scheduled and processed task
 1: Initialize:
 2: $Low\_Traffic\_FCs \leftarrow \emptyset$
 3: $High\_Traffic\_FCs \leftarrow \emptyset$
 4: Categorize FCs based on traffic:
 5: for each $fc \in FC$ do
 6:     if $fc.traffic\_level < threshold$ then
 7:         Add $fc$ to $Low\_Traffic\_FCs$
 8:     else
 9:         Add $fc$ to $High\_Traffic\_FCs$
10:     end if
11: end for
12: Task routing and processing:
13: if $T.deadline < deadline\_threshold$ then
14:     $fc \leftarrow SELECT(Low\_Traffic\_FCs)$
15:     $result \leftarrow IGEO\_PROCESS(T, fc)$
16: else
17:     $fc \leftarrow SELECT(High\_Traffic\_FCs)$
18:     $result \leftarrow RL\_PROCESS(T, fc)$
19: end if
20: return $result$
```

capacities and consumption rates. Various tasks from IoT devices were distributed among these nodes for simulation, with the task count varying from 200 to 600. Each processing node was equipped with a CPU that had a processing power ranging from 2000 to 6000 MIPS, and its energy usage while in operation was randomly assigned between 80 and 200 Joules to add diversity. Experimental results demonstrate that the proposed algorithms reduce response time in task scheduling, minimize total deadline violation time, and optimize resource and system energy consumption compared to other algorithms. The performance of IGEO on each of these parameters will be examined in the following sections.

A. Energy Consumption

Figure 4 illustrates the energy usage of FNs compared to task scheduling with varying task numbers (200 to 600) and 20 FNs. The energy consumption is determined using Eq. (5). The RIGEO approach is designed to optimize the energy consumption of FNs by effectively distributing tasks among nodes, thereby reducing overall energy usage. Comparing the IGEO method to other scheduling techniques reveals its success in minimizing energy consumption.

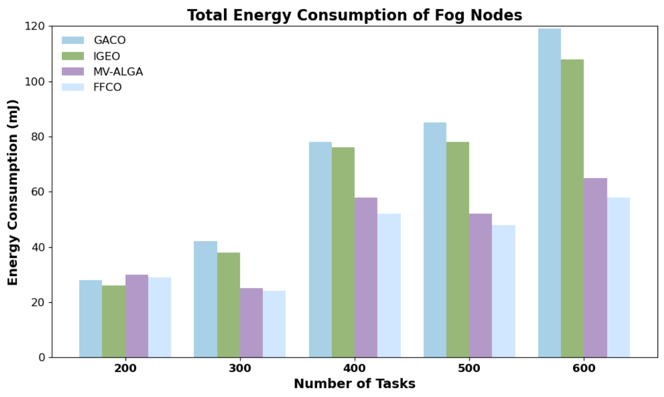

Fig. 4: Total Energy Consumption of Fog Nodes

B. Deadline Violation

The RIGEO technique has been proven to enhance adherence to deadlines compared to other methods, as evidenced by the decrease in deadline violation times for IoT tasks in Figure 5. The RIGEO method consistently chooses nodes with the shortest violation time, even in situations where there may be a shortage of nodes, demonstrating its flexibility and effectiveness in addressing deadline breaches. This capability is especially beneficial in real-world scenarios where adhering to deadlines is critical for achieving optimal performance and ensuring user satisfaction. In summary, the RIGEO method shows promise in enhancing system efficiency by minimizing deadline violations in scheduling, as evaluated using Eq. (3).

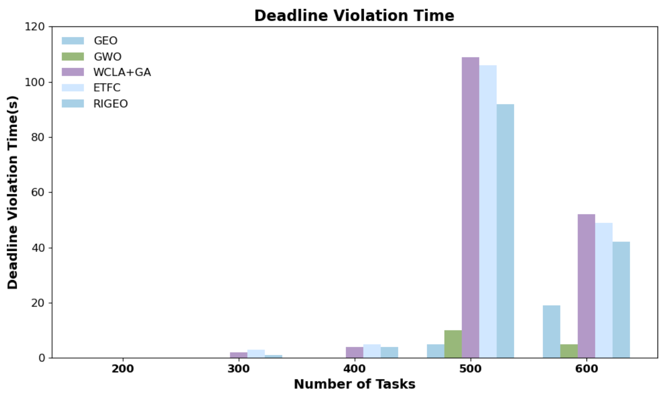

Fig. 5: Deadline violation time of FNs

C. Response time

Response time, defined as the period required for a system to respond to a given task, was a critical parameter in the proposed strategy and was systematically integrated into the fitness function. The results, as illustrated in Figure 6 with 20 FNs, highlight the superior performance of the RIGEO method in minimizing response time compared to other approaches. The computation of response time is detailed in Eq. (1).

IV. Conclusion

In recent years, the Internet of Things (IoT) has emerged as a pivotal technology in the internet and networking industry. However, IoT devices are constrained by limited resources, posing significant challenges. Fog computing has been introduced as a solution to address these limitations, yet integrating IoT with fog computing presents its own set of challenges,

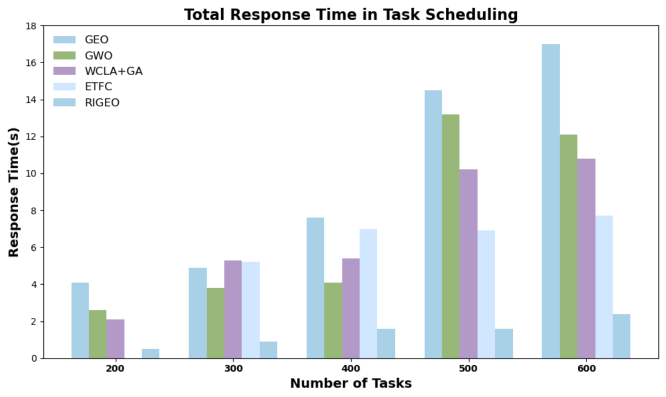

Fig. 6: Total response time of FNs

including resource management, energy efficiency, and adherence to deadlines for time-sensitive tasks. To tackle these issues, researchers have developed task scheduling solutions that primarily focus on optimizing either energy consumption, response time, or meeting strict deadlines.

In this work, a network is classified into low- and high-traffic based on the average traffic between network nodes. Additionally, the GEO algorithm is improved by incorporating genetic operators, resulting in the IGEO. IGEO utilizes genetic operators to enhance the exploration and exploitation of the search space. When network traffic is low, IGEO is used for task scheduling to ensure tasks are completed within their deadlines. When network traffic is high, the RL algorithm is utilized for task scheduling, as it offers better performance but has a higher time complexity. By using these algorithms based on network traffic levels, efficient task scheduling and optimization can be achieved in the network.

The experimental results demonstrate that the proposed algorithms significantly enhance the percentage of tasks meeting their deadlines, minimize the total time associated with deadline violations, and optimize resource energy consumption when compared to existing algorithms.

References

[1] M. Aazam, S. Zeadally, and D. Karras, "Deploying fog computing in industrial internet of things and industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4674–4682, 2020.

[2] M. Nematiollahi, A. Ghiffari, and A. Mirzaei, "Improved off-loading in internet of things based on the task multi-objective aquila optimizer," *Signal, Image and Video Processing*, vol. 1, pp. 545–552, 2021.

[3] H. Saadiah, H. Motameni, and R. Golsorkhtabaram, "Multi-objective environment-aware services placement optimization for environment using parallel ffd-genetic algorithm," *Pervasive and Mobile Computing*, vol. 92, p. 101800, 2023.

[4] H. Salehi Shayegan, A. Arakeri, and A. Salehi Shayegan, "Optimizing resources allocation for fog computing inah internet of things networks to reduce latencpycan," *Computational economics*, vol. 1, no. 3, pp. 99–112, 2024.

[5] K. Rezvani, A. Gaffari, and M. E. Dishabi, "The bedbug meta-heuristic algorithm to solve optimization problems," *Journal of Biomic Engineering*, vol. 20, no. 5, pp. 446–2485, 2020.

[6] M. Nematiollahi, A. Ghiffari, and A. Mirzaei, "An improved version of the moth-flame optimization algorithm," *Cluster Computing*, vol. 27, no. 2, pp. 1775–1797, 2023.

[7] M. Azami, J. Shojafar, J. Abawajy, and R. Buyya, "Deadline-aware and energy-efficient iot task scheduling in fog computing systems: A semi-greedy approach," *Journal of network and computer applications*, vol. 201, p. 103333, 2023.

[8] M. Seifhosseini, H. Shirvani, and Y. Ramzanpour, "Multi-objective cost-aware bag-of-tasks scheduling optimization model for iot applications running on heterogeneous fog environment," *Computer Networks*, vol. 240, p. 110161, 2024.

[9] S. Ghanavati, J. Abawajy, and D. Izadi, "Dynamic fault tolerant task scheduling approach in fog computing," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 1, pp. 488–499, 2023.

[10] F. Alqahtani, M. Amooh, and A. A. Nasr, "Scheduling and load balancing for requests in cloud-fog computing," *Peer-to-Peer Networking and Applications*, vol. 14, no. 6, pp. 1903–1916, 2021.

[11] P. Q. Gao, N. Di, D. J. Tri, E. N. Qhat, and E. N. Huh, "A performance-effective approach for task scheduling based on collaboration between cloud and fog computing," *International Journal of Distributed Sensor Networks*, vol. 18, no. 11, p. 1550147717742073, 2022.

[12] M. Khojand, K. Majidzadeh, and M. Y. Ghani, "Fairness controller placement in sdn using game theory and a discrete hybrid metaheuristic algorithm," *The Journal of Supercomputing*, vol. 80, no. 3, pp. 6552–6600, 2022.

[13] H. A. Rafiue, M. S. U. Shah, T. Islam, S. S. Maqsoud, C. M. Khan, and C. Maple, "A proposed nature-inspired hybrid algorithm (nbiha) for efficient resource management in fog computing," *IEEE Access*, vol. 7, pp. 1550–1573, 2019.

[14] M. A. Bagha, K. Majidzadeh, M. Masdari, and Y. Farhang, "Ela-rcp: An energy-efficient and load balanced algorithm for reliable controller placement in software-defined networks," *Journal of Network and Computer Applications*, vol. 225, p. 103855, 2022.

[15] Y. Wang, G. Kong, X. Zhang, X. Chen, C. Luo, and M. T. Tao, "Emef: Maximal energy efficient task scheduling in homogeneous fog networks," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4076–4087, 2018.

[16] S. D. Vispute and V. Vashisht, "Energy-efficient task scheduling in fog computing based on particle swarm optimization," *SN Computer Science*, vol. 4, no. 4, p. 591, 2023.

[17] M. Louail, M. Esseghir, and M. Merghem-Boulahia, "Dynamic task scheduling for fog nodes based on deadline constraints and task frequency in smart factories," in *2020 IEEE 6th International Conference on Network of the Future (NoF)*, pp. 16–22, IEEE, 2020.

[18] K. Klatoun, M. A. Matrouk, M. Mohammed, R. Nedoma, and J. M. Martinek, "A novel low-latency and energy-efficient task scheduling framework for internet of medical things in an edge fog cloud system," in *Sensors*, vol. 22, p. 5327, MDPI, 2022.

[19] M. Hosseini, N. Nickray, and S. Ghanbari, "Optimized task scheduling for cost-latency trade-off in mobile fog computing using fuzzy analytical hierarchy process," *Computer Networks*, vol. 206, p. 108752, 2022.

[20] N. Nikoukar, A. M. Balador, A. M. Rahmani, and H. Zakeri, "Task scheduling in fog-cloud environment," in *2020 CSI/CPSS International Symposium on Real-time and Embedded Systems and Technologies (RTEST)*, pp. 1–8, IEEE, 2020.

[21] S. Ghaffari and V. Mallothra, "A deadline-aware priority based semi-greedy task scheduling technique in fog computing," *in SC7*, vol. 2024, pp. 218–228, 2024.

[22] A. Seyyedabbasi, "A reinforcement learning-based metaheuristic algorithm for solving global optimization problems," *Advances in Engineering Software*, vol. 178, p. 103411, 2023.

[23] A. Mohammadi-Balani, M. Dehghan Nayeri, A. Azar, and M. Taghizadeh-Yazdi, "Golden eagle optimizer: A nature-inspired metaheuristic algorithm," *Computers & Industrial Engineering*, vol. 152, p. 107050, 2021.

[24] M. Taghizadeh-Yazdi, "Golden eagle optimizer: A nature-inspired metaheuristic algorithm," *Computers & Industrial Engineering*, vol. 152, p. 107050, 2021.

[25] N. Banes and A. K. Kang, "Cloud optimized task scheduling algorithm in cloud computing," in *Frontiers in Intelligent Computing: Theory and Applications: Proceedings of the 7th International Conference on Frontier Computing (FC 2020)*, pp. 1–9, Springer, 2022.

[26] S. D. Vispute and V. Vashisht, "Energy-efficient task scheduling in fog computing based on particle swarm optimization," *SN Computer Science*, vol. 4, no. 4, p. 591, 2023.