An Enhanced Grey Wolf Optimization Algorithm for Efficient Task Scheduling in Mobile Edge Computing

Jafar Aminu
Faculty of Computer Science and Information Technology University Putra Malaysia

Rohaya Latip
Faculty of Computer Science and Information Technology University Putra Malaysia

Zurina Mohd Hanafi
Faculty of Computer Science and Information Technology University Putra Malaysia

Shafinah Kamarudin
Faculty of Computer Science and Information Technology University Putra Malaysia

Bashar Umar Kangiwa
Department of Computer Science
Kebbi State University of Science and Technology Aleiro

Ayuba Liman
Department of Computer Science
Kebbi State University of Science and Technology Aleiro

ABSTRACT
Mobile edge computing (MEC) is a fundamental paradigm that brings computational resources closer to end users, minimizing latency and improving performance for real-time applications. Task scheduling optimization is generally a significant difficulty in MEC systems because of the dynamic nature of edge servers, limited processing resources, and energy restrictions. This leads to several problems, such as high energy consumption, makespan, extended task execution durations, and inefficient resource utilization. An enhanced grey wolf optimization method that introduces novel strategies to balance the exploration and exploitation processes more successfully will be used in this study to address these problems. The suggested EGWO algorithm handles the dynamic task allocation for maximum usage of resources, which minimizes makespan and energy consumption. We undertake comprehensive simulations for different workloads and show that EGWO consistently performs better than state-of-the-art techniques like WOA, PSO, and RFOAOA. EGWO leads to significant improvements in energy efficiency and makespan. It is, therefore a reliable and scalable solution for scheduling tasks in the MEC environment

Keywords
Mobile Edge computing, Task Scheduling Energy consumption, Makespan, Gray Wolf Optimization.

1. INTRODUCTION
Mobile Edge Computing (MEC) is an emerging computing paradigm that brings computational resources closer to end-users by integrating them with network edge devices, such as cellular base stations[1]. Unlike traditional Mobile Cloud Computing (MCC), where tasks are processed in distant data centers, MEC significantly reduces communication delays by performing computations at the network's edge. This proximity to end-users is crucial for applications requiring low latency and high throughput, such as augmented reality, innovative healthcare, and industrial IoT [2]**.** While MEC offers numerous advantages, such as reduced latency and improved quality of service (QoS), it also presents significant challenges in resource management due to the limited computational power, storage, and network resources of edge servers[3]. Efficient task scheduling in MEC is critical to managing these constraints while optimizing performance metrics such as latency, energy consumption, and resource utilization[4][5]. Task scheduling involves the assignment of computational tasks to available resources in a way that ensures minimal makespan (total task completion time), reduced energy consumption, and low operational costs [6]**.**

The core problem addressed in this paper is how to efficiently schedule tasks in a mobile edge computing environment to minimize makespan and energy consumption while dealing with edge servers' resource limitations and dynamic nature. Traditional algorithms often need help balancing exploration and exploitation, leading to suboptimal solutions in complex, real-time scenarios. In applications like augmented reality or competent healthcare, inefficient task scheduling can lead to significant delays, reduced quality of service, and excessive energy consumption, making real-time performance untenable for end-users[7].

In this context, traditional scheduling algorithms often need help keeping up with MEC environments' dynamic and resource-constrained nature. As a result, advanced optimization techniques have gained prominence in addressing these challenges[8]. Among them, metaheuristic algorithms have shown promise due to their ability to solve complex optimization problems by mimicking natural processes or behaviors. Specifically, the Grey Wolf Optimizer (GWO) algorithm, a relatively new metaheuristic approach inspired by the social hierarchy and hunting behavior of grey wolves, has emerged as a powerful solution [9]

The GWO algorithm has several key strengths, including a reduced number of search parameters, strong global search capabilities, and simplicity in implementation. These features make GWO particularly well-suited for dynamic and resource-limited environments like MEC, where computational efficiency and scalability are essential [10] However, despite its advantages, the standard GWO algorithm faces some limitations, such as slow convergence and a tendency to become trapped in local optima, especially in complex optimization scenarios.[11]. Existing methods such as Whale Optimization Algorithm (WOA), Particle Swarm Optimization (PSO), and Reinforced Fruit Fly Optimization Algorithm (RFOAOA), though effective in various optimization problems, often fail to adapt quickly enough to the rapidly

changing resource availability and task requirements in MEC environments, leading to suboptimal task allocation and high energy consumption.

To address these limitations, this paper introduces an Enhanced Grey Wolf Optimization (EGWO) algorithm that incorporates new strategies aimed at achieving a more effective balance between exploration and exploitation during the optimization process. The proposed EGWO algorithm is specifically designed to optimize task scheduling in MEC by improving resource utilization, reducing makespan, and minimizing energy consumption.

As MEC continues to evolve, with more real-time applications being deployed at the edge, solving the task scheduling problem efficiently is critical for enabling scalable, low-latency services that can meet the increasing demands of users. Through extensive simulations, we demonstrate that EGWO consistently outperforms existing task scheduling methods, including WOA, PSO, and RFOAOA. Our approach not only enhances the convergence rate but also ensures that the solution avoids premature convergence to local optima.

Main Contributions:

i. A task scheduling strategy is developed for edge computing environments to optimize both makespan and energy consumption.
ii. An EGWO-based method is introduced to address the task scheduling problem, with its performance evaluated through Python simulations. The results are compared against those obtained from WOA, PSO, and RFOAOA.

The rest of this paper is structured as follows: Section 2 offers a review of related work on task scheduling within edge computing environments. Section 3 outlines the problem formulation, and Section 4 details the developed model. Section 5 presents the simulation results, while Section 6 concludes the paper and suggests directions for future research.

2. RELATED WORKS

This paper [12] addressed the issue of task scheduling in edge computing (EC) through the incorporation of reinforcement learning techniques into representation models. The findings show that, in terms of energy consumption and service level agreement violation (SLAV), the suggested representation model in conjunction with a DRL-based algorithm performs better than the baseline techniques on average. The author [13] examines and determines whether the energy cost minimization problem is NP-hard while taking virtual machine migration, workload distribution, and green energy scheduling into account. A heuristic approach that approximates the ideal answer is suggested as a way to deal with computational complexity. It is shown through thorough simulations that the suggested algorithm may attain near ideal performance and cut brown energy usage significantly. The author [14] suggests a cooperative task scheduling method for edge computing supported by the Internet of Things. This method takes into account the completion of local tasks as well as the time of offloaded tasks for each IoT device. In contrast, an edge node selects the IoT devices that will carry out the tasks that have been offloaded by taking into account variables like energy consumption and execution time. The study [15] The approach initially assumes a system configuration with a single virtual machine (VM) and then extends to provide three heuristics for systems with multiple VMs. The proposed method, which utilizes immigrated VMs based on the Minimum Energy Transferring Attenuation Ratio (METAR), has proven effective in reducing overall energy consumption and brown energy usage, while simultaneously increasing the utilization of green energy, as evidenced by simulation results, the author [16] tackles the issue of multi-objective job scheduling in a 6G network with MEC support. The next task scheduling problem the paper addresses using an improved multi-objective cuckoo search (IMOCS) algorithm based on directed acyclic graphs (DAG) with the goal of lowering user equipment (UE) energy consumption and execution latency. This author [17] examines the job scheduling method in the context of fog computing. By using a task scheduling method based on a hybrid heuristic (HH) algorithm, the study primarily addresses the problem of terminal devices with low computing power and high energy consumption. This tactic makes it more feasible to process jobs for terminal devices more effectively and in real time. In the end, the experimental findings show that the suggested method performs better than alternative approaches.

3. SYSTEM MODEL

Figure 1 depicts the environment at the cloud's edge, structured into three layers: the user layer, edge layer, and cloud layer. The proposed EGWO algorithm is deployed at the edge layer, where it initially verifies the availability of resources on the MEC server to ensure they meet user requirements. If the resources are insufficient, the request is escalated to the cloud layer. Additionally, the MEC server's algorithm processes all users' requests, prioritizing them as needed. The MEC server comprises multiple servers, including virtual machines and mini datacenters, with the MEC server manager overseeing resources like processors and virtual machines (VMs). The user layer, at the bottom, sends requests to the nearest MEC server. The EGWO algorithm is then utilized to allocate tasks to the appropriate virtual machines (VMs) it identifies both underutilized and overutilized (VMs), deactivating the underutilized ones. This method significantly reduces energy consumption and shortens the makespan.

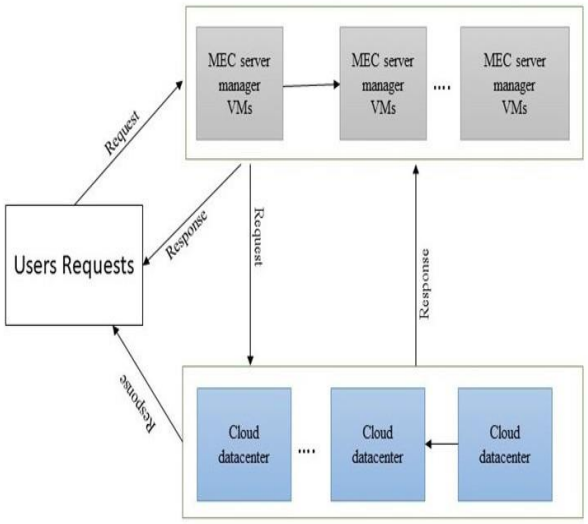

Fig 1 task scheduling model on edge computing

3.1 Problem Description

In an edge computing environment, the set of mobile devices is represented as $K = (k_1, k_2, \dots, k_n)$, while the set of MEC servers is denoted as $S = (s_1, s_2, \dots, s_m)$, and the set of virtual

machines (VMs) is identified as $V = (vm_1, vm_2, \dots, vm_p)$. These virtual machines are responsible for processing the tasks $T = (t_1, t_2, \dots, t_q)$ submitted by users. Each task t_i is characterized by its arrival time at_i, size or length len_i, deadline ddl_i, and completion time ct_i. A scheduler assigns these tasks to the appropriate vm_s vm_j with the goal of optimizing VM resource usage, reducing energy consumption E, minimizing and shortening the total task completion time (makespan). The aggregate processing time for all tasks P_{total} can be expressed as the cumulative sum of individual processing times P_{ij} across all VMs, represented by:.

$$P_{total} = \sum_{i=1}^{q} \sum_{j=1}^{p} P_{ij} \quad (1)$$

To address challenges such as prolonged task execution times, inefficient resource usage, and excessive energy consumption, we propose an enhanced grey wolf optimization (EGWO) approach. This method dynamically adjusts task allocation to optimize VM utilization, reduce makespan, and minimize energy consumption. The adaptive nature of the EGWO continuously improves the task scheduling process, leading to enhanced overall system performance in edge computing environments.

3.1.2 Energy Consumption Model

Energy $= E + (T_{current} - T_{last\ used}) \times P_{host}$ **(2)**

Energy consumption refers to the total energy utilized by the system, including the network's edge, sensors, gateway, and other components. This energy consumption can be calculated using Equation 2. To determine the energy usage of edge devices, the power consumption of each host is measured over a specific period, where EC represents energy consumption, $T_{current}$ is the current time, and $T_{last\ used}$ represent last utilization time, respectively. Once

the energy is calculated, resource management is carried out using the enhanced gray wolf optimization algorithm.

3.1.3 Maksepan

The term makespan refers to the total time required to complete all computational processes. Therefore, efficient job mapping is crucial to achieving a shorter makespan. The makespan, denoted as MKS, is calculated as follows.

Makespan $= max_k \in \{1,2, \dots, p\} \left(\sum_{j=1}^{q} Processing\ time_{jk}\right)$

4. PROPOSED APPROACH

The GWO algorithm is inspired by the social hunting strategies and leadership characteristics of grey wolves, which belong to the Canidae family. This algorithm operates in three phases: tracking, encircling, and attacking. As illustrated in **Fig. 2**, grey wolves are classified into four distinct types: α, β, δ, and ω. These types form a strict hierarchical social structure. The Alpha grey wolf holds the dominant position in the pack, establishing hunting and sleeping schedules that are adhered to by the other, more submissive wolves. The Beta wolf occupies the second tier in the hierarchy and aids the alpha in decision-making, enforcing the alpha's commands among the other wolves. When the alpha becomes old or dies, the beta is the most likely candidate to take its place. The delta wolf ranks third in the hierarchy, submitting to and taking orders from the alpha and beta wolves while exerting dominance over the omega wolves. The omega wolf is at the lowest level of the hierarchy, submitting to all the more dominant wolves above it, including the alpha, beta, and delta[18]. An interesting aspect of grey wolves' social behavior is their teamwork during hunts, which is influenced by this hierarchical structure. The primary hunting strategies employed by grey wolves are outlined below

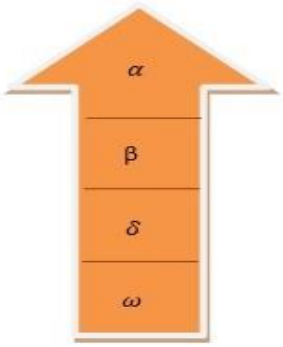

Fig 2. Hierarchy of grey wolf

4.1 Mathematical Model

The fittest wolves, α, β, and δ, are used to mathematically describe the social hunting behavior of grey wolves in order to find the best possible solution. In this hunting strategy, ω wolves follow behind the more dominant wolves.

4.2 Encircling the pray

grey wolves hunt by chasing and encircling their victim. It is modeled mathematically according to Eq. (4) and (5):

$$L = |Q \times Y_p(t) - Y(t) \quad (4)$$

$$Y(t+1) = Y_p(t) - B \times L \quad (5)$$

In this context, t represents the current iteration, and $t+1$ indicates the subsequent iteration. The prey's position vector is indicated by Y_p, while the position vector of the grey wolf is represented by Y. The coefficient vectors B and Q are expressed as follows:

$$B = 2 \times b \times r2 - b \quad (6)$$

$$Q = 2 \times r2 \quad (7)$$

Over the course of the iterations, the components of b reduce linearly from 2 to 0, while r1 and r2 are random vectors within the range of [0, 1]

4.3Hunting

Grey wolves use their α, β, and δ hunting mechanisms to guide them as they encircle their prey once they have been roughly located. The prey's location is unknown throughout the search space, but the α, β, and δ wolves are thought to be aware of it. As a result, other agents, also known as lesser gray wolves, constantly alter their own positions in accordance with the top three options that are retained.
The hunt is mathematically guided by equation (10), which is derived from equations (8) and (9).

$$L_\alpha = |Q_1 \times Y_\alpha - y|, L_\beta = |Q_2 \times Y_\beta - y|, L_\beta = |Q_2 \times Y_\beta - y| \quad (8)$$

$$Y_1 = Y_\alpha(t) - B_1 \times L_\alpha, Y_2 = Y_\beta(t) - B_2 \times L_2, Y_3 = Y_\delta(t) - B_3 \times L_\delta \quad (9)$$

$$Y(t+1) = Y_1 + Y_2 + Y_3/3 \quad (10)$$

The parameters b and Q are utilized to locate and attack prey, managing both exploration and exploitation. The parameter a, which decreases from 2 to 0, is designed to maintain a balance between exploration and exploitation, grey wolves spread out to search for prey, and when $|B|>$ 1and when $|B|<1$, they converge to initiate an attack. Incorporating randomness is beneficial in avoiding entrapment in local minima

4.4 Enhanced GWO

The current GWO algorithm provides an optimal solution that avoids local optima, yet there remains room for improvement in balancing exploration and exploitation. To enhance this, the Enhanced Grey Wolf Optimization (EGWO) method is introduced as a more effective algorithm. The EGWO algorithm incorporates several advancements aimed at improving both convergence rate and efficiency. In this method, the parameter a, which is a random vector within the range of [0, 1], is essential for maintaining the balance between exploration and exploitation. Adaptive values for the a parameter are used to ensure continuous exploration, thereby preventing the algorithm from becoming trapped in local optima and addressing issues related to accuracy. This parameter is critical for optimizing the solution vectors and can also be adjusted to control the algorithm's convergence rate. In emulating the hunting behavior of grey wolves, it is assumed that the fittest wolf, α, has the best knowledge of the prey's likely location, leading to the retention of only the most optimal solution. The global optimum is reached with the help of the population's best overall solution. The EGWO algorithm is specifically designed to improve performance in terms of accuracy, convergence rate, and avoiding premature convergence. The hunting mechanism is described using Equations (11–13).

$$L_\alpha = |Q_1 \times Y_\alpha - Y| \quad (11$$

$$Y_1 = Y_\alpha(t) - B_1 \times L_\alpha \quad (12$$

$$Y(t+1) = Y_1 \quad (13)$$

Below is a description of the suggested EGWO algorithm for task scheduling optimization issue.

Algorithm 1: The Enhanced Grey Wolf Optimizer (EGWO) for task scheduling

1. **Initialize** grey wolf positions (task allocations) and evaluate their fitness using Equations (1) and (2).
2. Identify α, β, and δ wolves (top three solutions).
3. Repeat until termination condition is met:
   a. **For** each grey wolf:
      i. Update position using α, β, and δ wolves' positions (refer to Equations (6), (7), (8), and (10)).
      ii. Apply dynamic adaptation to balance exploration and exploitation.
      iii. Recalculate fitness and update position if improved.
   b. Update α, β, and δ wolves.
4. **Return** the best solution found (optimal task scheduling configuration).

5. EXPERIMENTAL CONFIGURATION AND RESULT EVALUATION

To assess the performance of our proposed EGWO algorithm in comparison to PSO WOA and RFOAOA previous task scheduling method referenced in , we conduct a series of experiments in this section. The focus of the evaluation is on task makespan and energy consumption within a mobile edge computing (MEC) environment. In this scenario, various mobile devices generate distinct applications, each comprising multiple tasks that must be processed by edge resources. It is assumed that three MEC servers are connected to several mobile devices. Our simulations involve four sets of tasks, with totals of 300, 600, 900, 1200 and 1500 tasks. To account for inherent variability, the job lengths are randomly generated. The simulations are executed using Python 3.12 on a windows PC equipped with an intel Core i7 processor to validate the performance of our proposed approach.

5.1 Results

Table 1 shows the makespan results for different task sizes

ALGORITHMS	Task				
	300	600	900	1200	1500
WOA	95	180	301	379	499
PSO	90	177	300	379	498
RFOAOA	87	170	294	370	498
EGW	81	165	290	361	481

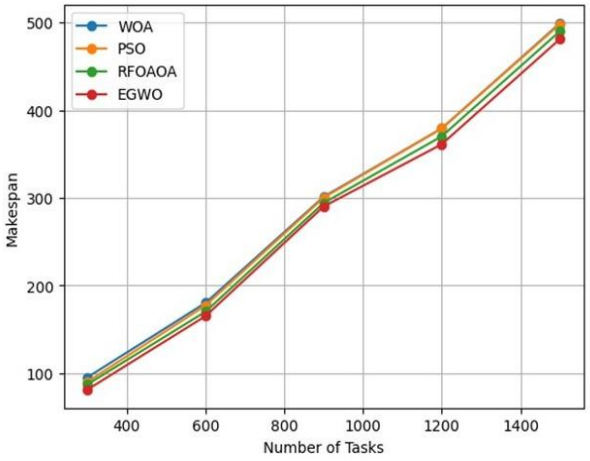

Fig 3 Comparison of Average Makespan

Table 2 shows the energy consumption results for different task

ALGORITHMS	Task				
	300	600	900	1200	1500
WOA	4.61×10^4	1.8×10^4	4.59×10^4	7.79×10^4	1.37×10^5
PSO	4.60×10^4	1.75×10^4	4.56×10^4	7.71×10^4	1.32×10^5
RFOAOA	4.59×10^4	1.75×10^4	4.47×10^4	7.69×10^4	1.29×10^5
EGW	4.49×10^3	1.75×10^4	4.46×10^4	7.62×10^4	1.21×10^5

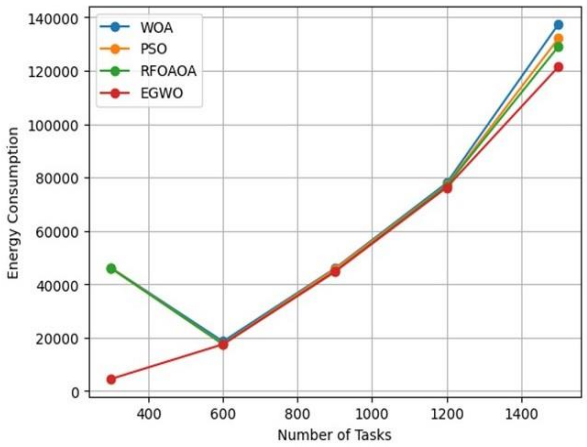

Fig 4 Comparison of Total Energy Consumption

5.1.2 Discussion of Results

Figure 3 compares the makespan performance of various task scheduling algorithms, including the proposed enhanced (EGWO), (WOA), (PSO), and (RFOAOA). In mobile edge computing (MEC) environments, where the timely processing of tasks is critical to maintaining quality of service (QoS) and ensuring a positive user experience, the makespan defined as the total time needed to complete all scheduled tasks serves as a vital performance metric. The results clearly show that the EGWO algorithm consistently achieves a shorter makespan across all task sets, ranging from 300 to 1500 tasks, when compared to the other algorithms. Notably, for the maximum load of 1500 tasks, the EGWO reduces the makespan to 481-time units, outperforming WOA, PSO, and RFOAOA, which register makespans of 499-, 498-, and 498-time units, respectively. This superior performance of EGWO can be largely attributed to its enhanced exploration and exploitation mechanisms, which effectively prevent premature convergence to suboptimal solutions. By dynamically adjusting the hunting strategy of the grey wolves within the algorithm, EGWO navigates the search space more efficiently, leading to better task allocation and faster task completion times. This makes EGWO particularly well-suited for real-time applications in MEC environments, where low latency is crucial. Furthermore, the EGWO's ability to maintain a low makespan as the number of tasks increases highlights its robustness and adaptability, key characteristics that are essential in the dynamic and resource-constrained environments typical of MEC. The consistent reduction in makespan directly enhances system performance, ensuring that end-user applications run smoothly without unnecessary delays.

Figure 4 presents a comparison of energy consumption among the EGWO, WOA, PSO, and RFOAOA algorithms across the same range of tasks. In MEC, where edge devices typically operate under strict power constraints, energy efficiency is a critical concern. Reducing energy consumption not only prolongs the operational life of edge resources but also supports the overall sustainability of the computing infrastructure. The EGWO algorithm shows a clear advantage in terms of energy consumption across all task scenarios. For example, when scheduling 1500 tasks, the EGWO algorithm reduces energy consumption to 1.21×10^5 units, significantly outperforming WOA, PSO, and RFOAOA, which consume 1.37×10^5, 1.32×10^5, and 1.29×10^5 units, respectively. This reduction in energy consumption can be credited to the EGWO's efficient resource allocation strategy, which actively identifies and deactivates underutilized virtual machines (VMs), thereby conserving energy. Additionally, the EGWO's improved convergence properties ensure that tasks are scheduled in a way that minimizes unnecessary energy use, avoiding the overhead associated with prolonged task processing and idle resource usage. The results suggest that the EGWO not only excels in reducing makespan but also significantly lowers the energy consumption of MEC systems. This dual optimization of time and energy is crucial for deploying scalable, efficient, and eco-friendly edge computing solutions, by minimizing both makespan and energy consumption, the EGWO enhances the overall operational efficiency of MEC environments, making it a highly effective tool for modern computing infrastructures that require both high performance and sustainability.

6. CONCLUSION

In this paper, we proposed an enhanced (EGWO) algorithm designed to address the critical challenges of task scheduling in mobile edge computing (MEC) environments. Our primary objective was to minimize both makespan and energy consumption, two pivotal factors that directly influence the efficiency and sustainability of MEC systems. Through comprehensive simulations, we demonstrated that the EGWO outperforms several established optimization algorithms, including (WOA), (PSO), and (RFOAOA). The results revealed that the EGWO consistently achieves a lower makespan across various task loads, thereby ensuring faster task completion times and enhanced quality of service (QoS). This reduction in makespan is particularly significant in real-time applications, where low latency is crucial for maintaining seamless user experiences.

7. CONFLICT OF INTEREST

There is no conflict of interest. All authors have agreed to its submission.

8. ACKNOWLEDGEMENT

This study is supported by the Universiti Putra Malaysia and the Ministry of Higher Education Malaysia under grant Number: (FRGS/1/2023/ICT11/UPM/02/3). We are sincerely grateful for the facilities and funding provided, which were essential for the completion and publication of this study.

9. REFERENCES

[1] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, and L. Guo, "Computation offloading in mobile edge computing networks: A survey," *J. Netw. Comput. Appl.*, vol. 202, no. April, p. 103366, 2022, doi: 10.1016/j.jnca.2022.103366.

[2] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 54–61, 2017, doi: 10.1109/MCOM.2017.1600863.

[3] M. E. C. Networks, "A Survey of Energy Optimization Approaches for Computational Task Offloading and Resource Allocation in," 2023.

[4] A. Mahjoubi, K. J. Grinnemo, and J. Taheri, "An Efficient Simulated Annealing-based Task Scheduling Technique for Task Offloading in a Mobile Edge Architecture," *Proc. 2022 IEEE Conf. Cloud Netw. 2022, CloudNet 2022*, pp. 159–167, 2022, doi: 10.1109/CloudNet55617.2022.9978900.

[5] A. Muhamad and M. Hussin, "Governing Resource Failures through Reinforcement Learning Scheduling in Fog/Edge Computing: A Review," *2024 IEEE Int. Conf.*

Autom. Control Intell. Syst. I2CACIS 2024 - Proc., no. June, pp. 256–261, 2024, doi: 10.1109/I2CACIS61270.2024.10649846.

[6] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A Survey on Mobile Augmented Reality with 5G Mobile Edge Computing: Architectures, Applications, and Technical Aspects," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 2, pp. 1160–1192, 2021, doi: 10.1109/COMST.2021.3061981.

[7] M. Sharma, A. Tomar, and A. Hazra, "Edge Computing for Industry 5.0: Fundamental, Applications, and Research Challenges," *IEEE Internet Things J.*, vol. 11, no. 11, pp. 19070–19093, 2024, doi: 10.1109/JIOT.2024.3359297.

[8] S. M. Altowaijri, "Workflow Scheduling and Offloading for Servicebased Applications in Hybrid Fog-Cloud Computing," *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 12, pp. 726–735, 2021, doi: 10.14569/IJACSA.2021.0121290.

[9] M. B. Gawali and S. K. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach," *J. Cloud Comput.*, vol. 7, no. 1, 2018, doi: 10.1186/s13677-018-0105-8.

[10] K. K. Mishra, "Grey Wolf Optimization," *Nature-Inspired Algorithms*, pp. 131–143, 2022, doi: 10.1201/9781003313649-6.

[11] D. Gabi *et al.*, "Dynamic scheduling of heterogeneous resources across mobile edge-cloud continuum using fruit fly-based simulated annealing optimization scheme," *Neural Comput. Appl.*, vol. 34, no. 16, pp. 14085–14105, 2022, doi: 10.1007/s00521-022-07260-y.

[12] Z. Tang, W. Jia, X. Zhou, W. Yang, and Y. You, "Representation and Reinforcement Learning for Task Scheduling in Edge Computing," *IEEE Trans. Big Data*, vol. 8, no. 3, pp. 795–808, 2022, doi: 10.1109/TBDATA.2020.2990558.

[13] L. Gu, J. Cai, D. Zeng, Y. Zhang, H. Jin, and W. Dai, "Energy efficient task allocation and energy scheduling in green energy powered edge computing," *Futur. Gener. Comput. Syst.*, vol. 95, pp. 89–99, 2019, doi: 10.1016/j.future.2018.12.062.

[14] Y. Kim, C. Song, H. Han, H. Jung, and S. Kang, "Collaborative Task Scheduling for IoT-Assisted Edge Computing," *IEEE Access*, vol. 8, pp. 216593–216606, 2020, doi: 10.1109/ACCESS.2020.3041872.

[15] Q. Zhang, X. Lin, Y. Hao, and J. Cao, "Energy-Aware Scheduling in Edge Computing Based on Energy Internet," *IEEE Access*, vol. 8, pp. 229052–229065, 2020, doi: 10.1109/ACCESS.2020.3044932.

[16] J. Li *et al.*, "Multiobjective Oriented Task Scheduling in Heterogeneous Mobile Edge Computing Networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 8, pp. 8955–8966, 2022, doi: 10.1109/TVT.2022.3174906.

[17] J. Wang and D. Li, "Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing," *Sensors (Switzerland)*, vol. 19, no. 5, 2019, doi: 10.3390/s19051023.

[18] N. Issa, Z. Alaa, and I. Abed, "Solving Suggested Problems using Grey Wolf Optimization," 2022, doi: 10.4108/eai.7-9-2021.2314893.