Systematic Review of Metaheuristic Task Scheduling Algorithms on Edge Computing

[1]Jafar Aminu [2,]Rohaya Latip, [3]Zurina Mohd Hanafi, [4]Shafinah Kamarudin, [5]Danlami Gabi

[1,2,,3,4]Faculty of Computer Science and Information Technology Universiti Putra Malaysia
[1,5]Department of Computer Science, Kebbi State University of Science and Technology Aleiro, Nigeria

Abstract
The proliferation of data sensors and mobile devices generates extensive amounts of data that must be processed in real-time. This requirement is fulfilled through nearby edge servers, though effective task scheduling is crucial to ensure timely task completion within resource limitations. With the increase in data volumes, timely task completion in MEC becomes an uphill task. Because of the ample solution space, task scheduling in edge computing is an NP-hard problem. While polynomial-time algorithms cannot provide optimal solutions, metaheuristic-based techniques have proven effective in near-optimal solutions within a short time frame. This paper presents a systematic review and comparative analysis of key metaheuristic algorithms, Particle Swarm Optimization (PSO), Genetic Algorithms (GA), and Ant Colony Optimization (ACO) in the context of task scheduling for MEC. The study evaluates these algorithms against key quality of service (QoS) parameters such as energy consumption, completion time, execution time, delay, and cost. A total of 103 studies were analyzed, and it has been found that though these metaheuristic techniques bring many advantages, key issues are still to be considered. Task relocation during scheduling needs much attention to be made more feasible. Furthermore, most of the current techniques need more consideration of network bandwidth, which leads to possible transmission breakdowns. In the end, the paper concludes that this could be a direction for future research, considering these factors and studying the integration of metaheuristics with AI techniques to develop more robust and adaptive scheduling strategies for MEC environments.

Keywords:
Mobile Edge Computing, Task scheduling Metaheuristic Approaches, Quality of service

1. Introduction

Mobile Edge Computing (MEC) is designed to operate near mobile devices, with moderately capable servers strategically positioned at the network edge to fulfil user-focused requirements and application-specific needs [1].This architecture supports a wide range of applications, including Internet of Things (IoT) systems and Augmented Reality (AR) technologies, which are gaining widespread adoption across various fields such as education, arts, manufacturing, entertainment [2],[3][, and e-learning platforms that significantly enhance learning experiences among others [4].

Despite the considerable advantages of Mobile Edge Computing (MEC), the task scheduling problem in these contexts is far from trivial. Huge search spaces can randomly generate satisfactory task assignments[5].

Scheduling algorithms take considerable time to converge, which can lead to the underutilization of virtual machines. This inefficiency circumscribes the maximum potential of edge computing systems and negatively affects the timely execution of tasks, especially in IoT and real-time data processing[6].

Moreover, task scheduling in edge computing is a critical process aimed at efficiently executing multiple tasks by utilizing the available computing resources to achieve optimal performance [7]. This process is guided by several optimization criteria, including energy consumption, completion time, execution time, reliability, Makespan, execution cost, budget, and resource utilization [8]. In the scheduling method, users submit tasks to an edge scheduler, which evaluates the status of resources through the edge information service. The scheduler then maps these tasks to the most suitable resources based on their specific requirements [9]. A well-designed scheduler ensures that tasks are assigned to resources such as virtual machines (VMs) in a manner that maximizes efficiency and minimizes resource wastage [10].

Despite its importance, task scheduling in edge computing is inherently a complex nondeterministic polynomial-time (NP) problem due to the complexities associated with resource management, task interdependencies, and the dynamic nature of edge environments [11]. While researchers have explored polynomial-time solutions, only some approaches have been universally successful in delivering optimal results across all scenarios. Meta-heuristics are preferred for edge computing task scheduling to overcome these limitations since they can accommodate complex optimization problems and show flexibility in adaptation to a dynamic environment[12].

Most of these metaheuristic algorithms perform well in balancing the multiple, often contradictory, quality-of-service (QoS) parameters important for tasks like energy consumption, task completion time, and

Manuscript received December 5, 2025
Manuscript revised December 20, 2025
https://doi.org/10.22937/IJCSNS.2025.25.12.22

resource utilization in edge computing [13]. Heir has already proven efficacy, combined with its flexibility in integrating into emerging techniques such as machine learning, thus ensuring that the metaheuristics approach will remain an influential and highly trustworthy choice in optimizing edge computing systems [14]. While all this progress has been made, the effectiveness of existing solutions still needs to be improved by the significant restriction on the aspects of Scalability and Adaptability in dynamic environments. Most of these algorithms face increasing difficulties in balancing multiple QoS parameters simultaneously, which will cause a tradeoff that can degrade the holistic performance of MEC systems [15]

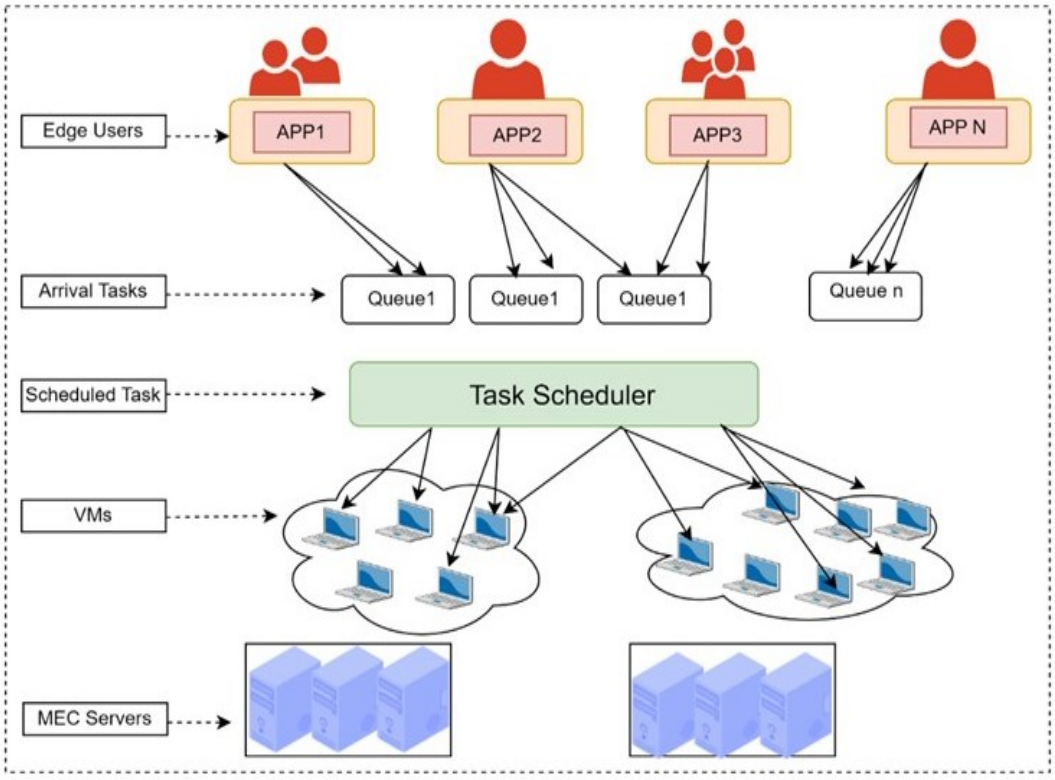


Figure 1: Task scheduling process

As shown in Figure 1, scheduling competing tasks on virtual machines within an edge computing environment is categorized as an NP-hard problem [16]. Optimization algorithms offer a promising solution to this challenge, but their complexity increases due to the dynamic nature of edge computing [13], [17]. Besides, comprehensive comparisons across different meta-heuristic approaches still need to be included, which makes it hard to pinpoint the best solution for a specific use case. This work, therefore, tries to compare the four pioneering metaheuristic algorithms, Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Ant Colony Optimization (ACO), comprehensively, especially for task scheduling in mobile edge computing. The study will, therefore, systematically assess these algorithms concerning different scenarios to clearly understand their respective strengths and weaknesses, which would subsequently inform the development of more effective and efficient scheduling strategies in MEC environments.

The structure of this paper is organized as follows: Section 1 provides the introduction, background, research contributions, and review questions. Section 2 outlines the research methodology. Section 3 delves into the task scheduling methods. Section 4 focuses on metaheuristic techniques. Finally, Section 5 discusses the results, and Section 6 concludes the paper.

1.1 Motivations

Mobile edge computing is fast emerging as a solution for the explosive growth of real-time data processing and reduced latency in distributed systems. Among the most critical challenges faced in mobile edge computing is metaheuristic task scheduling, which

optimizes key quality-of-service parameters such as energy consumption, task completion time, and latency. Thus, while numerous scheduling algorithms exist, there remains a significant knowledge gap regarding how these approaches compare, especially in dynamic and resource-constrained environments. This lack of comprehensive insight, on top of the increasing demand for more flexible and efficient scheduling solutions, justifies the current study's focus.

1.2 Contribution

The contributions of the study are as follows:

- This study thoroughly examined task scheduling techniques in the context of edge computing from 2015 to 2024, evaluating their pros and cons and the implementation tools used.
- An in-depth analysis of the literature on task scheduling was conducted.
- QoS Metrics Comparison: Various quality-of-service (QoS) metrics were compared, including utilization and percentage.
- This study identified future research challenges aimed at improving task scheduling methods in the edge computing context.

1.3 Review Questions

This study highlighted the below research questions:

RQ1: Which metaheuristic algorithm optimizes key QoS parameters in mobile edge computing (MEC) environments?

RQ2: How do these algorithms compare Scalability and Adaptability to dynamic MEC environments?

RQ3: What are the tradeoffs associated with using each algorithm, and how can these be mitigated to improve overall system performance?

RQ4: Can a hybrid approach combining metaheuristics with AI techniques offer superior performance compared to metaheuristics alone?

2. Methodology

A systematic literature review (SLR) is conducted to explore and categorize task scheduling techniques. An SLR typically involves detailing the research project's search strategy. Table 1 illustrates the performance of the SLR across scientific databases such as Google Scholar, Science Direct, Springer Link, and IEEE Xplore, along with their respective URLs. Figure 2 shows the temporal distribution of journal articles published by several recognized publishers. Figure 2 presents the filtration process of the research papers on task scheduling in an online context up to 2024, with most articles published between 2015 and 2024. and selected relevant documents for this review. Table 2 in Section 2 provides the inclusion and exclusion criteria. Our analysis focused on publications from 2015 to 2024.

Table 1: Database and their URLs

Search database	Web address
IEEE Xplore	https://ieeexplore-ieee-org
ScienceDirect	https://www.sciencedirect.com
Springer	https://link.springer.com
Google Scholar	https://scholar.google.com
ResearchGate	https://www.researchgate.net

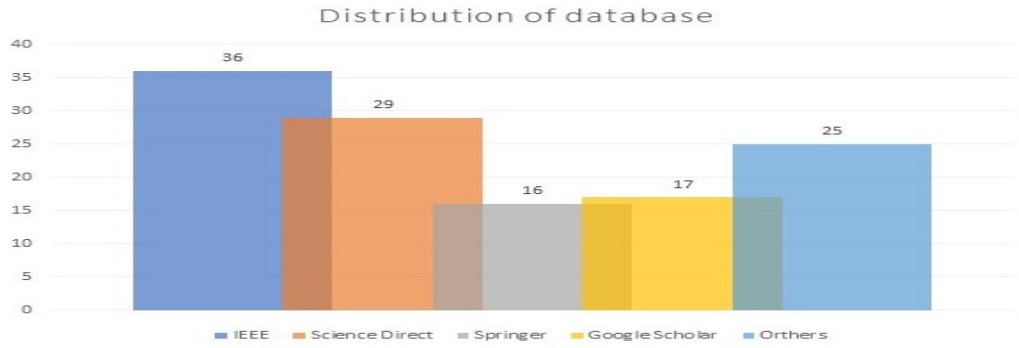


Figure 2 Distribution of the database

The search strategy facilitated the retrieval of relevant publications and enhanced our understanding of the concepts, terminologies, and keywords associated with mobile edge computing. Issues related to job scheduling methodologies aimed at reducing energy consumption were addressed. During the search and query process in research databases, specific terms and phrases were combined to locate pertinent publications. Consequently, the study used the matching search string of the library system engine: "((Task scheduling) AND Mobile edge computing) AND (genetic algorithms) AND (particle swarm optimization)." This approach yielded relevant papers that were downloaded for thorough evaluation.

2.2. Process of Study Selection

In this section, we analyzed the titles, abstracts, and conclusions of each paper retrieved from scientific databases during the initial search, which resulted in 850 papers. We selected the most relevant studies (n=103) and excluded the remaining studies (n=727) based on our research questions and their relevance to the study. The inclusion and exclusion criteria were applied to the 103 relevant papers. Figure 3 displays the PRISMA 2020 flow diagram, showing the number of documents obtained from the research database, as depicted in the PRISMA template. Each study found during the initial search was assessed before being included or excluded from the selected papers. The inclusion criteria are defined in Table 2

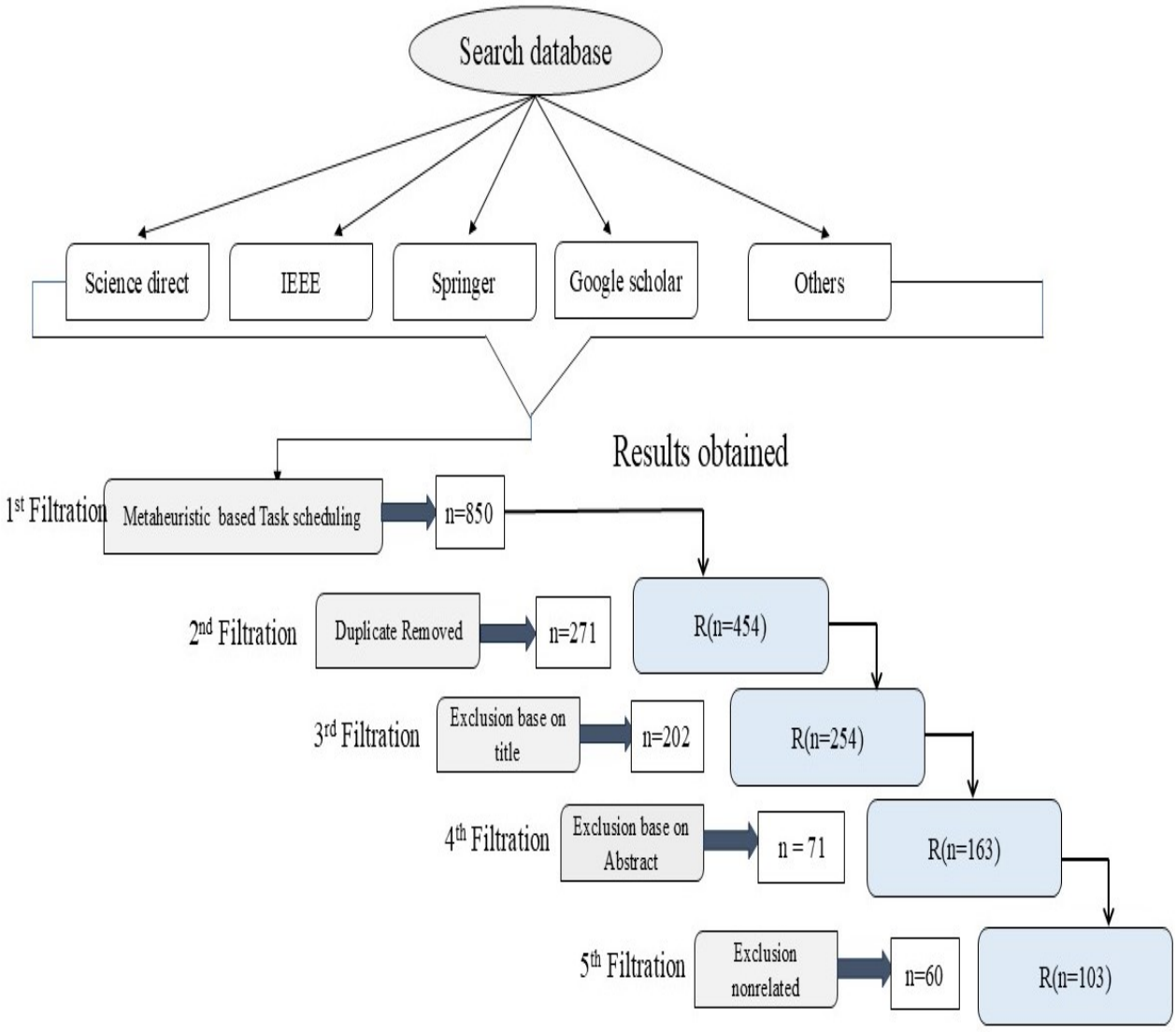


Figure 3: The filtration process

Table 2 Inclusion and Exclusion Criteria

SN	Inclusion	Exclusion
1	The article was published between 2015 and 2024.	Initially, we filtered, focusing on papers published from 2015 to the present. Following this filtration, the results narrowed down to (n=454) documents out of the initial (n=850 papers)
2	Studies on task scheduling techniques for edge computing	During the second filtering phase, specific papers were excluded based on their titles and keywords. As a result of this filtration, the number of selected documents decreased to (n=254)
3	The studies were published in English.	During the third round of filtration, exclusions were made based on the abstracts of the previously identified (n=254 papers), leading to a reduction to (n=163 papers).
4	Journal, review and conference	Ultimately, during the fourth filtration process, only research papers relevant to scheduling were selected for consideration in the proposed work, resulting in a total of (n=103 papers).

3. Background

3.1Task scheduling in MEC

Scheduling is the process of determining what tasks should be executed based on quality of service (QoS) factors[18]. Scheduling involves selecting the most suitable virtual machines for task execution using heuristic or metaheuristic algorithms while ensuring that quality of service (QoS) constraints are met[19]. According to network users, quality of service (QoS) pertains to the overall performance of a service within the edge network. This encompasses various parameters, such as deadlines, Makespan, cost, reliability, security, resource utilization, rescheduling, energy efficiency, and load balancing (see Figure 3). Below are brief descriptions of the measures commonly used in surveyed articles to quantify service quality.[20]

3.2 Scheduling Model

Scheduling involves assigning user tasks to available resources while meeting the constraints set by edge users and MEC service providers. Figure 4 illustrates the process of allocating various user tasks to available resources, particularly virtual machines (VMs) in edge servers.

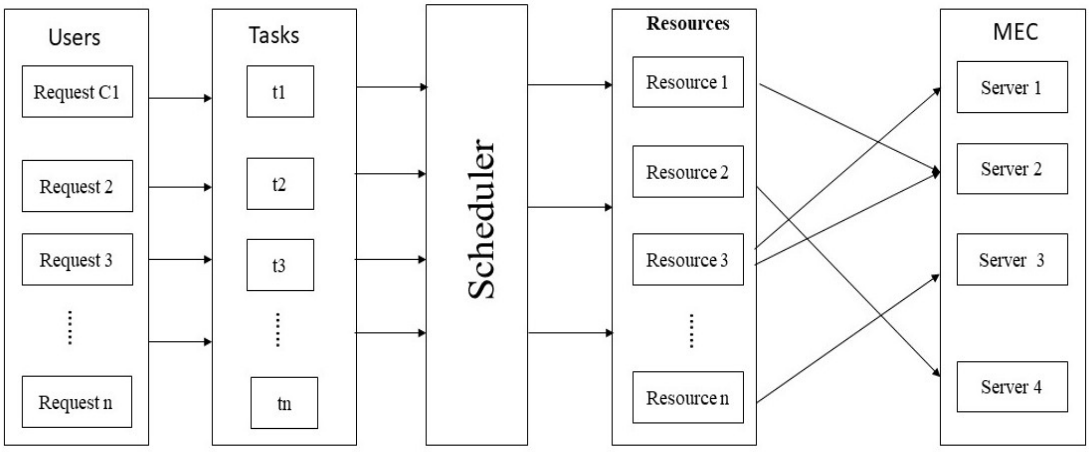


Figure 4 Task Scheduling Model

Tasks submitted by different users are independent and have varied requirements. Figure 4 illustrates that in a mobile edge computing context, there can be n requests ($request_1$, $request_2$..., $request_n$), each submitting n tasks (t_1, t_2..., t_n). These tasks are allocated to n resources ($resource_1$, $resource_2$..., $resource_n$) across multiple servers ($server_1$, $server_2$..., $server_n$).

3.3 Static and Dynamic Scheduling

3.3.1 Static Scheduling

Static scheduling requires having complete knowledge of all task and resource data before execution[21]. In a static approach, the scheduling algorithm establishes the allocation of tasks to resources before execution begins based on data from other units and the available resources[22]. Moreover, static schedules are crucial for predictable workloads and consistent resource performance. Careful design is essential for real-time, closed-loop, and mobility-sensitive systems. Some may opt for a static schedule to maintain predictability, avoid relying on mobile resources, and centralize decision-making within a single resource[23].

3.3.2. Dynamic Scheduling

Dynamic scheduling involves reorganizing tasks based on their priorities during execution. This method eliminates the need for preexisting information to complete tasks. Tasks are retrieved in real-time as they are executed[24]. Dynamic scheduling methods are designed to adapt to the evolving nature of tasks, which arrive at varying times depending on the status of system devices. These methods obscure workload data (tasks and nodes) and involve monitoring the target offloading node for optimal utilization in the edge context. The following authors have conducted research on scheduling using dynamic methods [25].

Table 3 reviews research works that combine heuristic and metaheuristic algorithms. Various proposed hybrid algorithms for scheduling are analyzed, distinguishing between dynamic and static scheduling types. The table also outlines the strengths and weaknesses of these algorithms and highlights the testing tools used in the experiments. Recent studies have focused predominantly on dynamic scheduling algorithms, with static scheduling having the lowest percentage. Additionally, hybrid scheduling algorithms are increasingly utilized, as indicated in Table 3, which provides a comparison of various studies focusing on task scheduling in edge computing environments, specifically evaluating the use of metaheuristic tools in static and dynamic contexts. Each study's tools, nature of scheduling (static or dynamic), advantages, and disadvantages are summarized as follows:

Table 3 Comparison of Metaheuristic Task Scheduling Approaches in Edge Computing

Ref	Tools	Static	Dynamic	Advantage	Disadvantage
[26]	MATLAB	No	YES	Significantly improves swarm convergence and load distribution compared to traditional methods.	Ignores constraints like deadlines or request priorities.
[27]	MATLAB	YES	NO	The method proved more effective, demonstrating improved energy efficiency and reduced delays.	It does not examine the overall system efficiency when using different algorithms.
[28]	NA	NO	YES	Enhancing the rate of convergence and search efficiency.	Not specifies
[29]	Python	NO	YES	Reduced task completion time and conserved energy.	It ignores real-world applications.
[30]	CloudSim	NO	YES	It provides a fast convergence rate and minimizes scheduling delays.	The successful quality of service needs to be evaluated.
[31]	NA	YES	NO	Significantly reduced the Makespan and energy consumption.	Variations in task length and transmission rates may occur due to environmental uncertainty.
[32]	Cloudsim	NO	YES	It enables users to select resources coming from various geographical locations while minimizing the duration of execution.	Not specified

In [27] presented three algorithms (BESO, PSO, and GA) for offloading computational tasks in MEC to enhance latency and effectiveness. BESO demonstrates superior energy utilization and reduced latency. This strategy has proven to be more effective, offering increased energy efficiency and fewer delays. [26] this study introduces the APDPSO algorithm for static load balancing, addressing issues in PSO-based methods by utilizing an additional archive and a novel discretization mechanism. The simulation results indicate that the proposed technique significantly improves swarm convergence, diversity, and load balancing compared to traditional methods. [28] addresses creating a unique region partitioning method for satellite communication swarm job scheduling for global monitoring, with a minor overflow frequency while optimizing satellite communication consumption. This study presents EA-DFPSO, a technique for power-efficient task scheduling in mobile edge networks that optimizes both energy conservation and job completion time. It also incorporates an additional fitness function to enhance energy savings in edge servers while maintaining the task quality of service, outperforming conventional scheduling techniques in edge computing contexts. Allocating and scheduling dynamic and virtual resources to clients for optimal revenue remains a significant challenge in edge environments.

According to [29], a multiobjective resource allocation algorithm, PSO/GA, has been proposed to reduce costs, delays, and energy consumption. This approach employs metaheuristic techniques to address scheduling challenges in edge environments. This study [32] uses two metaheuristic algorithms to address resource scheduling issues in the casting sector: particle swarm optimization for large-scale optimization and genetic algorithms for dynamic problems. This study adopts a hybrid strategy that combines genetic algorithms with particle swarm optimization to achieve global convergence at an optimal cost value. [30] method introduced by the author exemplifies this approach and includes five processes.

The initial step uses a task nature-oriented class neural network to classify incoming requests as sensitive or nonsensitive based on characteristics such as login information, email identification, passwords, resource variations, and performance parameters. This causes a decrease in scheduling delays. [31] presented two particle swarm optimization (PSO) methods that address the NP-hard problem with reduced speed. Specifically, a position-based mapping system links particles with scheduling solutions. This mapping strategy leverages both the current best solution and a position-based probability model to generate high-quality solutions that can inherit beneficial patterns from the current best solution. Table 4 provides a comparative analysis of various Quality of Service (QoS) parameters employed in metaheuristic task scheduling within edge computing environments, highlighting their effectiveness in optimizing performance and resource management.

Table 4 Comparison of quality-of-service parameters for static and dynamic task scheduling techniques

Ref	Energy consumption	Completion time	Execution time	Delay	Cost	Makespan	Scheduling Strategy	environment
[33]	✓	X	X	X	X	✓	Dynamic	MEC
[34]	X	X	X	✓	X	X	Dynamic	MEC
[35]	✓	X	X	X	X	X	Static	MEC
[36]	✓	X	✓	X	X	X	Dynamic	FMEC
[37]	X	X	X	✓	✓	X	Static	MEC
[38]	✓	X	✓	X	X	X	Dynamic	MEC
[39]	X	X	X	✓	X	X	Static	FMEC
[40]	✓	X	X	X	✓	✓	Dynamic	FMEC
[41]	✓	X	X	✓	X	✓	Static	FMEC
[42]	X	X	X	X	X	X	Static	MEC
[43]	✓	✓	X	X	X	X	Dynamic	MEC
[44]	✓	✓	X	X	X	X	Dynamic	MEC
[45]	X	X	✓	X	✓	✓	Dynamic	MEC
[46]	✓	✓	X	X	X	X	Dynamic	MEC
[47]	✓	X	X	X	✓	X	Dynamic	FMEC

Ref	Energy consumption	Completion time	Execution time	Delay	Cost	Makespan	Scheduling Strategy	environment
[48]	X	✓	X	X	✓	X	Static	FMEC
[49]	X	✓	✓	X	X	X	Static	MEC
[50]	✓	X	X	✓	X	X	Dynamic	MEC
[51]	X	X	X	X	X	X	Static	MEC
[52]	✓	X	✓	X	X	X	Dynamic	MEC
[53]	✓	X	X	X	X	X	Static	MEC
[54]	X	X	✓	X	X	X	Static	MEC
[55]	X	✓	X	X	X	✓	Dynamic	MEC
[56]	✓	X	X	X	X	✓	Static	MEC
[57]	✓	✓	X	X	X	X	Dynamic	MEC
[58]	X	✓	X	X	X	X	static	MEC
[59]	X	X	X	X	✓	✓	Dynamic	MEC

[43] proposed a dynamic and energy-efficient scheduling method for heterogeneous edge computing systems. This technique incorporates traffic mapping and proportional reallocation of energy slack space to minimize schedule length within energy constraints. It is particularly advantageous for delay-sensitive mobile applications, such as augmented and virtual reality, which handle large volumes of multimodal input. This study introduces an innovative strategy for joint task scheduling and resource allocation using a multi-action and environment-adaptive proximal policy optimization algorithm. This method improves existing research by incorporating task prioritization and dynamic transmit power allocation, resulting in faster completion times and reduced energy consumption [44]. This study introduces a fault-tolerant computational offloading and task-scheduling strategy designed to minimize redundancy. It encompasses three main procedures: offloading decisions, task scheduling, and redundancy minimization. The offloading decision algorithm identifies the optimal layer for task execution. Furthermore, the primary backup task scheduling method prioritizes execution time, energy consumption, CPU utilization, and dependability when scheduling tasks across edge and cloud environments[38] Similarly,

[33] proposed a dynamic semi-online scheduling technique for edge computing platforms based on the mapping. This study examines three key parameters influencing energy consumption on these platforms: job execution, transmission, and idle state. By considering the processing speed, routing delay, and queuing delay of edge nodes, which correspond to these critical factors, an energy consumption optimization problem is formulated. The proposed solution efficiently allocates system resources and reduces energy consumption in edge computing platforms within a semi-online framework [34]. The author formulated a constrained optimization problem in a cloud-edge environment to address a challenging task scheduling issue for edge device applications with priority constraints. The objective is to minimize energy consumption while adhering to mixed deadline requirements [37]. The author presented a task scheduling technique to reduce ad hoc mobile edge computing overhead. This method optimizes resource utilization, energy consumption, time delay, and monetary costs to minimize inefficiency for edge devices [36]. It introduces a task scheduling solution for industrial edge devices in edge computing using ant colony optimization. This algorithm optimizes task scheduling by evaluating execution time, network utilization, and energy consumption, drawing parallels to ant behaviour.

A scheduling technique was proposed in [35]. Tasks are divided and assigned using a task queue, with the optimal server for each application selected through an auction-bid process. The experimental results suggest that minimizing energy consumption depends on various factors, including the application release frequency, server quantity, and deadline constraints. In [45], the author proposed a semi-dynamic real-time task scheduling approach optimized for edge environments. This method effectively assigns tasks while minimizing energy consumption, costs, and Makespan. Additionally, the author employed a modified grey wolf optimizer (GWO) to optimize task allocation based on criteria such as workload duration, resource demand, and execution time. Evaluations demonstrated that GWO enhances job scheduling performance, providing an efficient solution for real-time computing challenges [46]. To achieve energy efficiency, the author developed a task scheduling technique utilizing dynamic voltage and frequency scaling (DVFS) technology combined with time-deterministic cyclic scheduling. This research explores a method capable of effectively adjusting processor states through dynamic frequency and voltage scaling, making it ideal for scheduling tasks across multiple processors. Unlike previous studies, this approach allows tasks to be assigned

to idle slots at lower voltage and frequency levels without violating dependency requirements or significantly extending the maximum completion time. [47] discussed scheduling precedence-constrained tasks for a smartphone application in a fog computing system.

The subsequent section focuses on the quality of service (QoS) metrics used in the examined research to analyze work scheduling algorithms. Figure 5 illustrates the evaluation rate of each QoS metric. Additionally, the review highlights the limitations identified.

4. Metaheuristic-Based Scheduling Techniques

Meta-heuristic algorithms have become indispensable for optimizing task scheduling in edge computing environments. Their capability to identify near-optimal solutions in complex and dynamic settings makes them highly effective in tackling resource management challenges[60]. For example, the Hybrid Meta-Heuristics Optimization Algorithm, which combines Genetic Algorithms (GA) and Particle Swarm Optimization (PSO), has demonstrated notable advancements in energy efficiency, resource utilization, and response times by efficiently allocating user task requests among edge devices and MEC servers[61]. Similarly, nature-inspired algorithms such as GA, PSO, and Ant Colony Optimization (ACO) have been evaluated in edge computing, with GA excelling in minimizing energy consumption and PSO excelling in reducing average flow time [62]. In cloud computing, meta-heuristic techniques like GA and PSO have been extensively studied to optimize task allocation by minimizing resource consumption, including memory usage, processing time, and computational costs[63]. Recent innovations have introduced advanced meta-heuristic methods, such as the. The Quantum-Behaved Particle Swarm Optimization (QPSO) algorithm has also proven highly effective in edge computing, demonstrating significant improvements in energy consumption, execution cost, and latency by efficiently allocating processing elements to tasks[64]. Comparative analyses of PSO, GA, and ACO have highlighted their effectiveness in optimizing energy consumption and execution time. Furthermore, in distributed stream processing systems (DSPS), meta-heuristic algorithms have surpassed conventional approaches by simultaneously meeting multiple Quality of Service (QoS) objectives. These findings emphasize the Adaptability and efficiency of meta-heuristic algorithms in optimizing task scheduling in edge computing environments, enabling better performance and enhanced resource management.

4.1. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) was introduced by Eberhart and John F. Kennedy in 1995. The behaviour of herds of animals, flocks of birds, and swarming fish inspired the PSO concept. The PSO algorithm is used in many scientific domains and serves as the foundation for many metaheuristic techniques because of its straightforward structure and possible computational complexity. Birds' social behaviour can inform computational techniques, like PSO, that address optimization issues.[65]. Every bird stands for a potential fix to choose the best course of action that maximizes the current circumstance; the particle swarm explores the area in the designated directions. PSO can reach faster convergence than evolutionary algorithms, which is a clear advantage.[66]. The velocity vector determines both the movement of a particle within the search space and how the particles are updated during the optimization process [67]. The particle's position and velocity are recalculated during each iteration using Equations 1 and 2.

$$x_i(t=+1) = x_i(t) + v_i(t) \quad (1)$$

$$v_i(t+1) = w \times v_i(t) + r_1 r_2 (x_i(t) - x_i(t) + r_2 r_2 (x(t) - x_i(t))$$

Here, w represents the inertia weight, and $r_{1\ and} r_2$ is a randomly generated number. The term x_i indicates the best-known position of the particle, $v_i(t)$ represents the particle's current velocity, and $x_i(t)$ corresponds to its current position.

Figure 5 presents the flowchart of the Particle Swarm Optimization (PSO) algorithm, providing a clear depiction of the entire process. It outlines each step, from initialization to velocity updates, fitness evaluation, and optimization, offering a systematic representation of the algorithm's workflow.

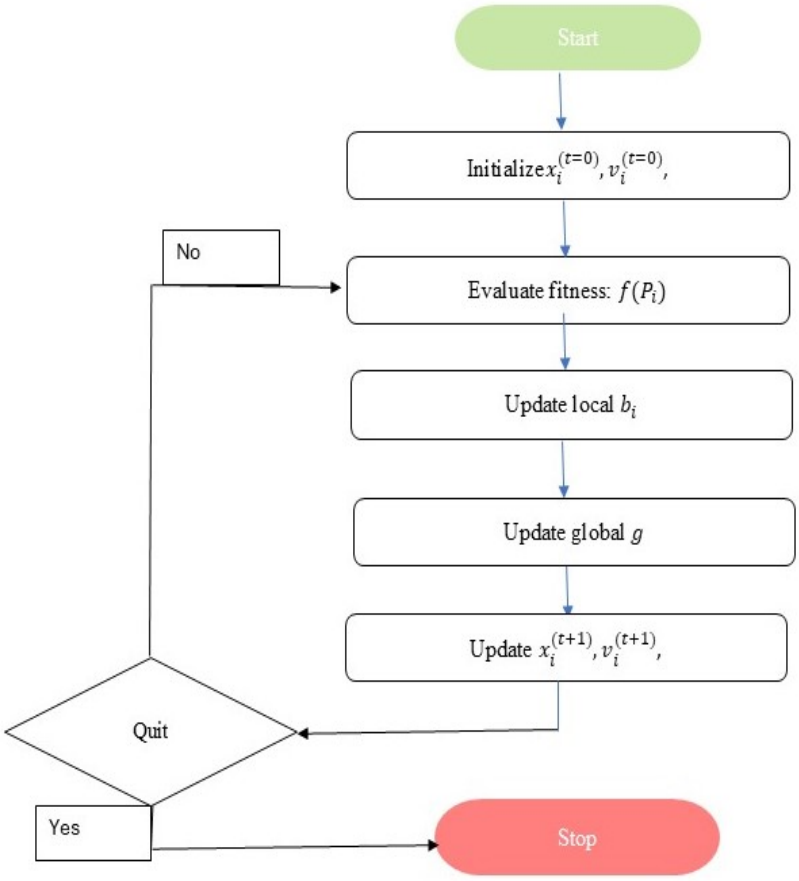


Table 1 Fig 5. Flow chart For PSO Algorithms

4.1.1 Overview of PSO-based scheduling mechanisms

The Particle Swarm Optimization (PSO) algorithm is widely recognized for its effectiveness in task scheduling within edge computing environments, thanks to its ability to efficiently solve complex optimization problems [68]. Over time, various modifications and enhancements have been introduced to improve its performance further in this context. Drawing inspiration from the social behaviours of animals like birds and fish, these advancements allow the PSO algorithm to adapt seamlessly to the dynamic and flexible demands of edge computing. Consequently, it delivers energy-efficient and cost-effective solutions while adhering to strict Quality of Service (QoS) requirements.

Recent studies have introduced innovative approaches to address optimization challenges in mobile computing environments. A notable example is a two-level scheduling framework designed to enhance efficiency. At the upper level, the Inertia Weight Particle Swarm Optimization (IW-PSO) method is utilized to model and solve mobile users' complex, non-convex, and nonlinear power allocation problem, primarily focusing on minimizing transmission energy consumption. At the lower level, the binary Particle Swarm Optimization (PSO) algorithm tackles the joint task offloading and resource allocation problem, formulated as a Mixed-Integer Nonlinear Programming (MINP) model. This hierarchical strategy provides an optimal scheduling solution that reduces energy consumption, improves response times, and enhances overall system performance [69].

This study introduces a hybrid method integrating Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) to optimize Mult objective job scheduling in fog computing environments. This approach outperforms traditional single-algorithm methods by harnessing the complementary strengths of both algorithms. The hybrid model balances exploration and exploitation of the search space, enabling more effective and efficient solutions. Experimental findings reveal that the proposed hybrid algorithm substantially enhances execution time and reduces energy consumption [70].

This research presents a heuristic particle swarm optimization (PSO) algorithm grounded in the Lyapunov

framework called LPSO. The algorithm is designed to maintain queue stability and ensure task completion within predetermined timeframes while balancing computational energy consumption at IoT nodes, transmission energy usage, and energy utilization within the fog computing environment. By optimizing these factors, the LPSO algorithm effectively minimizes total energy consumption and facilitates efficient task execution [71].

This research presents a Particle Swarm Optimization (PSO)-based method designed to lower energy costs and accelerate finding optimal solutions. Experimental findings demonstrate that the energy-efficient PSO metaheuristic harnesses its potential to deliver notable performance enhancements [72]. The main goal of scheduling is to optimize task completion time while minimizing energy consumption on mobile devices. To address this NP-hard challenge, this study introduces two optimization techniques based on slow-motion particle swarm algorithms. A key component of this approach is developing a position-based mapping strategy that converts particle positions into practical and feasible scheduling solutions [31].

Table 5 highlights recent studies that have employed the PSO algorithm to address task scheduling challenges across various use cases, including vehicles, the Internet of Things (IoT), the Industrial Internet of Things (IoT), and mobile devices (MDs).

Table 5: PSO Metaheuristic Algorithms for Task Scheduling

Ref	Main problem	Algorithm	Tools	Strength	Drawback
[73]	Based on resource availability, decide whether the task should be executed locally on a device or offloaded to the edge.	PSO	Python	Archived task scheduling issues efficiently, which lead to substantial efficiency enhancements.	Complexity in computational processes
[74]	It tries to minimize task execution time and scheduling costs by enhancing a task scheduling strategy.	PSO	Cloudsim	Effectively decreases task execution time, maximum completion time, and overall task scheduling cost in edge computing environments.	Insufficient real-world experimentation.
[75]	Optimizing time and cost in complex resource allocation scenarios	PSO	NA	Successfully improved PSO in lowering Makespan and cost, revealing their benefit and effectiveness in workflow scheduling.	There is a need to thoroughly investigate the DNCPSO method's ability to scale large-scale workflow applications. Complexity in computational processes
[31]	Decrease task completion time and energy consumption for mobile devices.	PSO	NA	Utilize efficient task ordering policies to generate optimal ini tial solutions	The techniques offered may not be suitable for handling the problem of many antennas on mobile devices.
[76]	to reduce the total duration of execution and cost related to dependent tasks in an edge computing environment.	PSO/WOL	Python	Archived task scheduling issues efficiently, which lead to substantial efficiency enhancements.	The suggested method is based on a more straightforward model and requires further evaluation in more complex situations.
[77]	Lower the cost and time of conducting dependent tasks in edge computing environments.	PSO/GWO	Cloudsim	Reduced execution time and cost as compared to PSO and GWO methods.	The workload has yet to be considered.
[78]	Optimizing time and cost in complex resource allocation scenarios	PSO	WorkflowSim	Successfully improved PSO in lowering Makespan and cost, revealing their benefit and effectiveness in workflow scheduling.	needs to thoroughly investigate the Scalability of the DNCPSO method for large-scale workflow applications.

Ref	Main problem	Algorithm	Tools	Strength	Drawback
[79]	Examine applications with accurate execution durations and computing costs.	PSO/DS	NA	It improves cost-effectiveness and timeliness. More precisely, those who possess limited resources can achieve effective rates above	PSO/DS requires significant financial resources to be implemented by users.
[80]	Network distribution of tasks and applications.	PSO/SA	Matlab	Using a hybrid method improves efficiency.	Latency issues have yet to be handled.
[81]	Reduce energy consumption across several processors in real-time embedded systems.	PSO	Python	Optimization effectiveness and performance improved significantly, allowing energy consumption to be more effective than the prior technique.	Concentrated primarily on energy consumption.
[82]	Efficient in terms of energy and secure data transmission	APSO	Cloudsim	Findings show that the method surpasses four advanced methods.	Need to Improve network lifetime with proper data collecting.
[83]	Scheduling workflows with several objectives on substantial distributed networks	GMPSO	NA	Results were substantial compared to well-performed techniques.	energy consumption needs to be coEsidere
[84]	limited wireless capacity and battery power.	PSO/GA	Python	Reduces energy consumption for competent edge providers and outperforms two leading evaluation techniques.	Hybrids of PSO and GA suffer from dealing with equality restrictions.
[85]	Lowering execution time and resource utilization to enhance load balancing.	FMPSO	Python	Minimize Makespan and improve effectiveness and degree of imbalance.	It needs to consider the priority of tasks and the distribution of load.
[86]	Efficient energy consumption and secure data transmission	PSO	Python	Minimize execution delay and energy consumption.	The model could not detect the link between workload and demand for resources.
[87]		PSO/WOA	Python		PSO and WOA need help in addressing equality limitations.
[88]	To decrease execution costs and time to meet the timeline.	BPSO	Maatlab	optimize cost-effectiveness and utilization of energy.	Decrease the adequate performance of the technique as the number of tasks increases.

5. GA base Task scheduling algorithms

Genetic Algorithms (GAs) have proven to be a powerful approach for addressing the complex challenges of task scheduling in diverse computing environments, including mobile edge computing (MEC) and cloud computing. Renowned for their efficiency, GAs are well-suited for identifying high-quality solutions within practical time limits, even for computationally intensive NP-complete problems like task scheduling. In this algorithm, parent selection is based on the fitness of each chromosome. This means that chromosomes with higher fitness levels are more likely to be chosen. The probability for each chromosome iii is calculated using the following formula:

$$P_i = \left(\frac{f_i}{\sum_{i=}^{n} f_j}\right) \quad (3)$$

Here, fif_ifi represents the fitness of chromosome iii and indicates the total population size. In recent years, Genetic Algorithms (GA) have been extensively utilized to optimize task scheduling challenges in both edge and cloud computing environments. Figure 6 depicts a flowchart outlining the process of using the genetic algorithm for task offloading. It provides a detailed overview of the entire operational cycle, including

initialization, selection, crossover, mutation, and convergence phases.

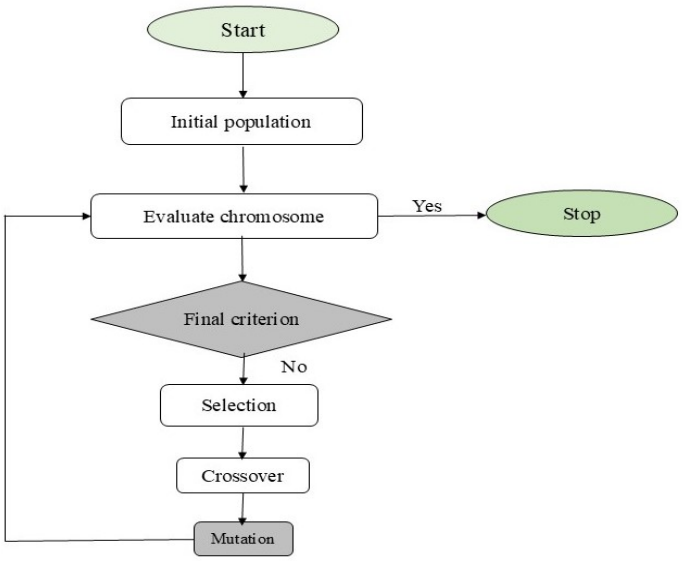


Figure 6 Flowchart of the GA

5.1 Overview of GA-based scheduling mechanisms

In mobile edge computing environments, the genetic algorithm (GA) metaheuristic offers a valuable approach to tackling complex scheduling and optimization challenges. The process begins with identifying the optimal allocation of resources, such as memory and processing power, on edge servers to execute specific applications while minimizing latency efficiently. This strategy involves defining a search space of potential solutions, which includes key factors such as network architecture, task scheduling, and resource allocation [89]. Once these components are established, the GA optimization process can be implemented using a suitable library or framework. Typically, this involves generating an initial population of solutions, iteratively refining them based on evaluation results, and progressively improving the population with each iteration. Recently, there have been some researchers who have applied (GA) task scheduling for the optimization of edge computing. This paper presents a mathematical model addressing the task scheduling challenge, aiming to minimize the total energy consumption of fog nodes (FNs) while ensuring that IoT tasks meet their Quality of Service (QoS) requirements [90]. This paper presents a task scheduling strategy called IGA-TSPA, which leverages an improved genetic algorithm. The proposed method enhances the initialization and mutation processes of the genetic algorithm, efficiently narrowing the initial solution space and expediting convergence to optimal solutions. A scheduling framework was designed to break down applications into individual tasks, which are subsequently organized in a task queue. An auction-based bidding mechanism was utilized to identify the most appropriate server for handling each task [91] and presented a unique hybrid model for task scheduling that considers both independent task limitations and server capacity restrictions. CBASGA develops in a structured and systematic manner within a highly adaptable initial population [92]. Using accurate cluster trace data from Alibaba, experimental results demonstrate that the proposed algorithm more effectively reduces work completion time than baseline approaches across various scenarios [93]. The author proposed two genetic algorithms incorporating heuristic concepts to schedule and allocate processor tasks. These algorithms aim to minimize execution time while maintaining load balance. Rigorous benchmark evaluations demonstrate that the proposed algorithms consistently outperform existing algorithms, highlighting their practical utility and effectiveness in real-world scenarios. The author focused

on static task scheduling in heterogeneous computing systems, which may restrict the applicability of the proposed genetic-based algorithm in dynamic task scheduling scenarios where tasks and system conditions evolve over time. The author utilizes a genetic algorithm to arrange tasks efficiently. This research seeks to optimize task scheduling in mobile-edge computing, aiming to reduce latency and offloading failures. The results closely approximate the ideal solution obtained through extensive analysis [94]. The author presented a scheduling approach that considers the varying resource requirements of activities and dynamically allocates resources based on their characteristics to enhance system performance. By emphasizing energy efficiency, this article contributes to the long-term sustainability of edge-computing systems, which are known for their resource limitations. [95] The author introduces a work scheduling strategy to reduce task delays and enhance task completion on edge devices. The proposed optimization approach, which is based on the genetic algorithm (GA), encodes tasks from edge devices in binary form, refines the selection function, and effectively minimizes task delays [96]. Table 6 presents several recent studies that have explored computational offloading on edge servers for various use cases using the GA algorithm.

Ref	Methos	Tools	Advantage	Disadvantage
[92]	GA	N/A	Studies indicate that CBASGA reduces the average job completion time by at least 21.7% over the baselines, which shows the efficiency of the stated method.	The dynamic changes in task priority are not thoroughly addressed.
[93]	GA	C++	The new algorithms regularly outperformed existing algorithms, demonstrating their effectiveness in actual applications.	The framework of optimization relies on a single-parameter solution, which impacts efficiency.
[97]	GA	Python	The model confirmed the technique's efficacy. The outcomes of these experiments demonstrated notable efficiency increases compared to alternative task scheduling systems.	The discussion of larger and more complex systems needs to be improved.
[98]	GA	Python	Simulation results show that the proposed algorithm provides performance close to the optimal solution, which is obtained through an exhaustive search	lengthy and intricate computations
[99]	GA	Matlab	Compared to other optimization strategies, the suggested HGA performs better in terms of the quality of the solution's performance.	Effectiveness is uncertain because of the estimated method's inherent unpredictability.
[94]	GA	Matlab	Decreased energy usage, which enhanced system performance and user experience by carrying out tasks effectively.	Fails to address how network delay affects scheduling effectiveness, a critical issue in edge computing situations.
[95]	GA	N/A	The study presents experimental results showing that, compared to alternative approaches, the suggested method significantly reduces task execution time and delays.	The local optimal problem and the meta-heuristic algorithm's parameters are complex to change.
[96]	GA	Python	Based on research findings, the PGA performs, on average, better in user satisfaction and resource efficiency than several recent works—by 27.9–65.4% and 33.8–69.6%, respectively.	Complexity in computational processes
[100]	GA	Python	A critical contribution of PEGA is its efficiency in producing near-optimal schedules through thorough search inside the search space, significantly improving total system performance.	NA
[101]	GA	NA	Reduced the time spent on calculation overall, saved energy and increased the proportion of tasks completed by the deadline.	lengthy and intricate computations
[102]	GA	Matlab	The simulation results showed that the suggested solutions offer performance nearly identical to the ideal one while requiring less complexity.	NA
[103]	QGA	Matlab	Regarding schedule quality, the suggested QGA algorithm performs better than a random search technique and two non-volutionary heuristics.	Low focus on data security and privacy

Table6:GA Metaheuristic Algorithms for Task Scheduling

5.2 Ant Colony

The Ant Colony Optimization (ACO) algorithm is a metaheuristic inspired by the natural behaviour of ants, specifically their use of pheromones to communicate and determine the most efficient routes between their colony and a food source. This phenomenon, referred to as staggered [104], forms the conceptual basis of the algorithm. In distributed computing environments such as edge, fog, and cloud systems, the number of ants employed is equal to or less than the number of tasks requiring scheduling. During the initialization phase, each ant is randomly assigned a task and resource for execution. Following this, the probability function is applied to determine the allocation of tasks to resources.

$$P_{(ij)} = \frac{\tau_{(ij)}^{\alpha} \times P_{(ij)}^{\beta}}{\sum \tau_{(ij)}^{\alpha} \times P_{(ij)}^{\beta}} \quad (4)$$

The pheromone value reflects the relationship between a task and its corresponding resource, while the heuristic function directs the allocation process. The impact of the pheromone value and the regulation of the heuristic function collectively influence their roles in guiding the decision-making process.

5.2.1.1Overview of ACO-based scheduling mechanisms

The Ant Colony Optimization (ACO) algorithm has proven highly effective in addressing task offloading challenges across various fields, including edge computing and construction project management. Drawing inspiration from ants' natural behaviour, the algorithm mimics their method of leaving pheromone trails to guide others [105]. These pheromone trails enable ants to navigate the solution space, marking their routes and assessing the quality of potential solutions to converge on the optimal one.

Recently, research has been conducted on ant colonis. The ACO technique was proposed in [106] for resource and task selection criteria in clusters[106]. To attain the required quality for scheduling workflows, users define QoS restrictions. [107] presents a workflow scheduling method based on the ant colony system (ACS) that has been improved with new features. The idea of a knowledge matrix combined with the ACO algorithm is covered in [108]. Researchers used the ACO technique to monitor the historical attractiveness of assigning jobs to the same physical machine. An energy-efficient scheduling system based on ant colonies was created, and pheromone schemes have been updated, as demonstrated in [109] and [110]. The author utilizes a genetic algorithm to arrange tasks efficiently. This research seeks to optimize task scheduling in mobile-edge computing to reduce latency and offloading failures. The results closely approximate the ideal solution obtained through extensive analysis

The author presented a scheduling approach that considers the varying resource requirements of activities and dynamically allocates resources based on their characteristics to enhance system performance. By emphasizing energy efficiency, this article contributes to the long-term sustainability of edge-computing systems, which are known for their resource limitations[94]. The author introduces a work scheduling strategy to reduce task delays and enhance task completion on edge devices [95]. Based on the genetic algorithm (GA), the proposed optimisation approach encodes tasks from edge devices in binary form, refines the selection function, and effectively minimizes task delays [96].

Table 7 offers a succinct overview of various research efforts utilizing Ant Colony Optimization (ACO) methods for task scheduling within Mobile Edge Computing (MEC) environments. The studies highlight a range of performance metrics, task types, and methodologies, illustrating the flexibility and effectiveness of ACO in enhancing MEC system optimization.

Table 7: ACO Metaheuristic Algorithms for Task Scheduling

Ref	Method	Performance metric	Nature of task	Environment
[111]	Ant Colony Framework utilizing various types of ant agents	Energy usage, constraints for Throughput, and Response Time	Independent	MEC
[112]	Utilizing a local search method on the results generated by ACO	Energy consumption and Makespan	Independent	MEC
[113]	Ants update a unified result set rather than modifying individual solutions	Load Balancing	Workflow	MEC
[114]	ACO Heuristics are chosen according to pheromone level	Delay and cost	Workflow	MEC
[115]	Hot spots are detected, and their load is redistributed using ACO	Energy consumption	Independent	MEC

[116]	Incorporating a knowledge model with ACO	Execution time and cost	Workflow	MEC
[117]	Incorporating vector algebra into ACO	Energy and resource utilization	Independent	MEC

6. Discussion and Comparison

This section provides an analytical analysis and discussion of the current tsk scheduling techniques in mobile edge computing based on metaheuristics.This systematic review evaluates key metaheuristic algorithms for task scheduling in mobile edge computing (MEC), focusing on Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Ant Colony Optimization (ACO). The analysis highlights their strengths, limitations, and performance based on key quality-of-service (QoS) parameters such as energy consumption, task completion time, execution time, delay, cost, and Makespan.

6.1 Classification for Task scheduling techniques

Figure 7 presents a statistical comparison of metaheuristic-based techniques applied in the MEC context, categorized according to the proposed taxonomy. This taxonomy highlights three primary approaches: Particle Swarm Optimization, Genetic Algorithms, and Ant Colony Optimization. As depicted in the figure, Particle Swarm Optimization accounts for most reviewed studies, representing 47.2%. Genetic Algorithms follow with a coverage rate of 33.3%, while Ant Colony Optimization occupies the remaining 19.4%.

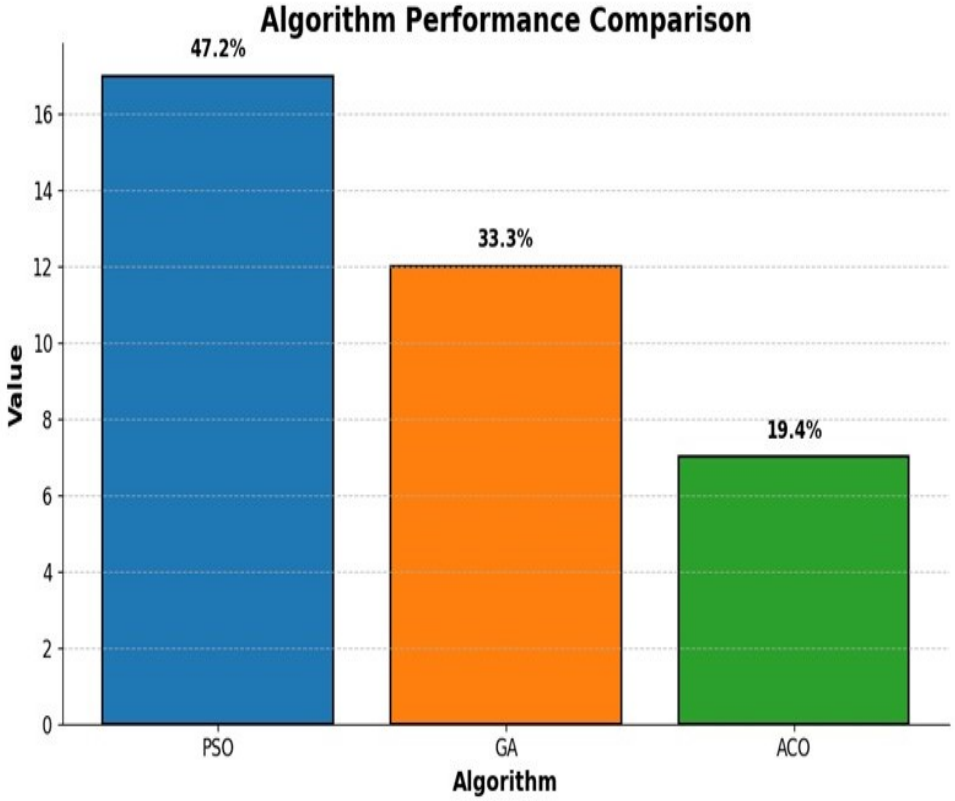


Figure 7 Classification of Metaheuristics-based techniques

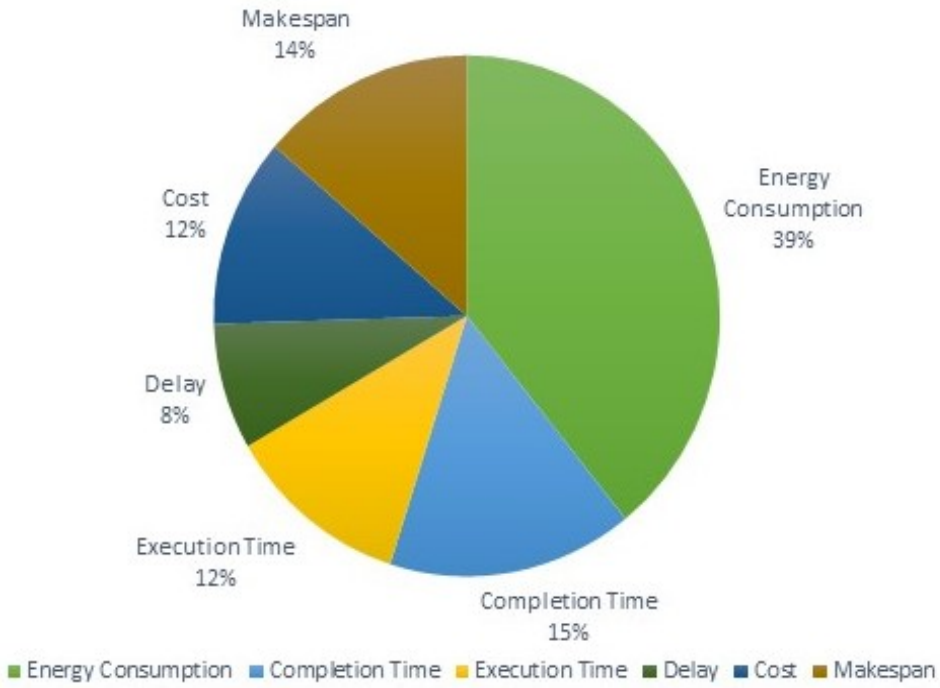


Figure 8 Comparison Performance Metric

According to the data in Figure 8, energy consumption is the most frequently used evaluation measure by researchers in the reviewed literature for assessing task scheduling techniques, with 39% of the recommended strategies focusing on minimizing energy usage. Completion time is the second most common metric, representing 15% of the usage, followed by Makespan at 14%. The cost and execution time restriction metrics account for 12% of the task scheduling rates. Specifically, 8% of the studies analyzed used cost as a criterion for evaluating task scheduling.

6.2 Comparison of Three Metaheuristic Scheduling Algorithms

Based on the analysis presented in the paper, the strengths of Particle Swarm Optimization (PSO) in comparison to other metaheuristic algorithms for task scheduling in mobile edge computing (MEC) are outlined below:

6.2.1 Energy consumption

PSO outperforms both GA and ACO in the aspect of energy consumption optimization. As a result of fast convergence and dynamic task allocation mechanism, it is very effective at minimizing energy consumption in diversified workloads. Though GA assured energy efficiency, yet because of computational complexity it may lead to higher energy cost in large-scale systems. ACOs have yet to be explored for energy-related optimizations compared to PSOs.

6.2.2 Task Completion Time

PSO has substantial reductions in the time taken for execution and completion of tasks because of the balance between exploration and exploitation. Variants such as Inertia Weight PSO improve these strengths further. GA also effectively reduces task delays but is sometimes slower converging compared to PSO because it depends on population-based strategies. ACO gave promising results for interdependent tasks, but due to iterative pheromone updating, it is less efficient concerning dynamic, real-time scheduling of tasks.

6.2.3 Scalability

The simplicity of PSO makes it scalable to moderately complex task scheduling problems. However, its performance may degrade in large-scale scenarios due to increased computational complexity. Such is the similar scalability issue with GA and ACO, since GA requires more computational resources for better performance, and convergence in ACO depends on many iterations.

6.2.4 Adaptability

PSO is adaptable to dynamic environments since the approaches quickly adapt when the workloads or available resources change. On the other hand, though

adaptive, the GA might show premature convergence in the case of highly dynamic scenarios. ACO does not adhere very well due to its static pheromone learning

6. 3 Future Research Direction

Future research should look into task scheduling strategies based on metaheuristics for a variety of problems:

i. Introducing novel metaheuristics for scheduling is essential, particularly in multi-server systems where selecting and allocating tasks to the appropriate server is crucial in optimizing key performance metrics.

ii. In the MEC environment, specific applications must operate in real-time due to the critical nature of scenarios, such as healthcare systems or automotive networks, which demand high data rates. However, these scenarios often need historical data for reference. As a result, there is a pressing need for innovative metaheuristic-based or hybrid approaches to ensure seamless and efficient scheduling of these vital applications.

iii. Integrate network factors such as bandwidth, latency, and transmission reliability into task scheduling models. This will enhance the practical applicability of scheduling algorithms in real-world scenarios where network conditions significantly impact system performance.

iv. Explore hybrid optimization strategies that simultaneously address computational and communication constraints for latency-sensitive applications.

v. Mobility poses a significant challenge in various research domains, such as Vehicular hoc networks (VANETs), Intelligent Transportation Systems (ITS), and Unmanned Aerial Vehicles (UAVs). However, despite its importance, mobility issues have been largely overlooked in the literature on Mobile Edge Computing (MEC). To address this gap, exploring advanced metaheuristic techniques combined with other mathematical models for more effective solutions is crucial.

7. Conclusion

This work systematically reviews three metaheuristics techniques, namely PSO, GA, and ACO, for task scheduling in MEC. The work validates PSO as the most powerful technique in optimizing energy consumption and task completion time, while GA and ACO are competent in multiobjective optimization and interdependent task workflows. However, scalability and adaptability issues remain significant in MEC's dynamic environments. For these limitations, hybrid methodologies in future research, which can incorporate metaheuristics with AI, would be better in terms of adaptability and scalability. These models can leverage the predictive power of AI combined with the optimization power of metaheuristics toward the development of strong and efficient scheduling solutions. Incorporating network-specific factors such as latency and bandwidth will add significant practicality to these approaches. It concludes the idea of scheduling MEC tasks and provides a platform for innovative approaches to be further developed to meet modern applications' demands.

Conflict Of Interest

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

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.

(The authors acknowledge that this paper was developed with the assistance of AI tools. Specifically, ChatGPT was used to enhance the readability of the information and polish arguments. Grammarly was used to improve the text's cohesion and clarity and to check for grammar. These technologies also ensured originality and correct referencing throughout the article, which helped reduce the chance of accidental plagiarism.)

References

[1] T. M. Ho and K. K. Nguyen, “Joint Server Selection, Cooperative Offloading and Handover in Multi-Access Edge Computing Wireless Network: A Deep Reinforcement Learning Approach,” *IEEE Trans. Mob. Comput.*, vol. 21, no. 7, pp. 2421–2435, 2022, doi: 10.1109/TMC.2020.3043736.

[2] S. K. Jagatheesaperumal, K. Ahmad, A. Al-Fuqaha, and J. Qadir, “Advancing Education Through Extended Reality and Internet of Everything Enabled Metaverses: Applications, Challenges, and Open Issues,” *IEEE Trans. Learn. Technol.*, vol. 17, pp. 1120–1139, 2024, doi: 10.1109/TLT.2024.3358859.

[3] H. M. Reeve, A. M. Mescher, and A. F. Emery,

"Experimental and numerical investigation of polymer preform heating," *Am. Soc. Mech. Eng. Heat Transf. Div. HTD*, vol. 369, no. 6, pp. 321–332, 2001, doi: 10.1115/imece2001/htd-24365.

[4] A. A. Almodaires, F. M. Almutairi, and T. E. A. Almsaud, "Pre-Service Teachers' Perceptions of the Effectiveness of Microsoft Teams for Remote Learning," *Int. Educ. Stud.*, vol. 14, no. 9, p. 108, 2021, doi: 10.5539/ies.v14n9p108.

[5] L. T. Hsieh, H. Liu, Y. Guo, and R. Gazda, "Deep Reinforcement Learning-Based Task Assignment for Cooperative Mobile Edge Computing," *IEEE Trans. Mob. Comput.*, vol. 23, no. 4, pp. 3156–3171, 2024, doi: 10.1109/TMC.2023.3270242.

[6] X. Zhang, Z. Cao, and W. Dong, "Overview of Edge Computing in the Agricultural Internet of Things: Key Technologies, Applications, Challenges," *IEEE Access*, vol. 8, pp. 141748–141761, 2020, doi: 10.1109/ACCESS.2020.3013005.

[7] Q. Luo, S. Hu, C. Li, S. Member, G. Li, and W. Shi, "Resource Scheduling in Edge Computing : A Survey," vol. 48202, pp. 1–36, 2008.

[8] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic Task Offloading and Scheduling for Low-Latency IoT Services in Multi-Access Edge Computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 668–682, 2019, doi: 10.1109/JSAC.2019.2894306.

[9] R. Alsurdeh, R. N. Calheiros, K. M. Matawie, and B. Javadi, "Hybrid Workflow Provisioning and Scheduling on Cooperative Edge Cloud Computing," *2021 IEEE/ACM 21st Int. Symp. Clust. Cloud Internet Comput.*, pp. 445–454, 2021, doi: 10.1109/CCGrid51090.2021.00054.

[10] K. Zhu, Z. Zhang, S. Zeadally, S. Member, and F. Sun, "Learning to Optimize Workflow Scheduling for an Edge – Cloud Computing Environment," *IEEE Trans. Cloud Comput.*, vol. 12, no. 3, pp. 897–912, 2024, doi: 10.1109/TCC.2024.3408006.

[11] Z. Bahroun and R. As, "THE MULTI-SKILLED RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM : A SYSTEMATIC REVIEW AND AN EXPLORATION OF FUTURE LANDSCAPES," vol. 0, 2024, doi: 10.2478/mspe-2024-0012.

[12] R. Aron and A. Abraham, "Engineering Applications of Artificial Intelligence Resource scheduling methods for cloud computing environment : The role of meta-heuristics and artificial intelligence," vol. 116, no. August, 2022.

[13] T. Alfakih, M. M. Hassan, and M. Al-Razgan, "Multi-Objective Accelerated Particle Swarm Optimization with Dynamic Programing Technique for Resource Allocation in Mobile Edge Computing," *IEEE Access*, vol. 9, pp. 167503–167520, 2021, doi: 10.1109/ACCESS.2021.3134941.

[14] H. Singh, S. Tyagi, P. Kumar, S. S. Gill, and R. Buyya, "Metaheuristics for Scheduling of Heterogeneous Tasks in Cloud Computing Environments : Analysis , Performance Evaluation , and Future Directions," vol. 1, no. 1, 2021.

[15] H. Hu, W. Song, Q. Wang, S. Member, and S. Y. Feb, "Energy Efficiency and Delay Tradeoff in an MEC-Enabled Mobile IoT Network," pp. 1–14.

[16] L. E. I. Yang, D. Yang, and J. Cao, "QoS Guaranteed Resource Allocation for Live Virtual Machine Migration in Edge Clouds," pp. 78441–78451, 2020, doi: 10.1109/ACCESS.2020.2989154.

[17] X. Wang, S. Member, Y. Han, S. Member, and V. C. M. Leung, "Convergence of Edge Computing and Deep Learning : A Comprehensive Survey," pp. 1–36.

[18] T. Zhu, T. Shi, J. Li, Z. Cai, and X. Zhou, "Task scheduling in deadline-aware mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4854–4866, 2019, doi: 10.1109/JIOT.2018.2874954.

[19] P. Singh, M. Dutta, and N. Aggarwal, "A review of task scheduling based on meta-heuristics approach in cloud computing," *Knowl. Inf. Syst.*, no. July, 2017, doi: 10.1007/s10115-017-1044-2.

[20] M. Haghi Kashani, A. M. Rahmani, and N. Jafari Navimipour, "Quality of service-aware approaches in fog computing," *Int. J. Commun. Syst.*, vol. 33, no. 8, pp. 1–34, 2020, doi: 10.1002/dac.4340.

[21] M. Padole and A. Shah, "Comparative study of scheduling algorithms in heterogeneous distributed computing systems," *Adv. Intell. Syst. Comput.*, vol. 562, pp. 111–122, 2018, doi: 10.1007/978-981-10-4603-2_12.

[22] P. Varshney and Y. Simmhan, "Characterizing application scheduling on edge, fog, and cloud computing resources," *Softw. - Pract. Exp.*, vol. 50, no. 5, pp. 558–595, 2020, doi: 10.1002/spe.2699.

[23] K. Sadeghi, A. Banerjee, J. Sohankar, and S. K. S. Gupta, "Optimization of Brain Mobile Interface Applications Using IoT," *Proc. - 23rd IEEE Int. Conf. High Perform. Comput. HiPC 2016*, no. Study 00000445, pp. 32–41, 2017, doi: 10.1109/HiPC.2016.014.

[24] L. Zhou, L. Zhang, B. R. Sarker, Y. Laili, and L. Ren, "An event-triggered dynamic scheduling method for randomly arriving tasks in cloud manufacturing," *Int. J. Comput. Integr. Manuf.*, vol. 31, no. 3, pp. 318–333, 2018, doi: 10.1080/0951192X.2017.1413252.

[25] M. K. Pandit, R. N. Mir, and M. A. Chishti, "Adaptive task scheduling in IoT using reinforcement learning," *Int. J. Intell. Comput. Cybern.*, vol. 13, no. 3, pp. 261–282, 2020, doi: 10.1108/IJICC-03-2020-0021.

[26] Z. Miao, P. Yong, Y. Mei, Y. Quanjun, and X. Xu, "A discrete PSO-based static load balancing algorithm for distributed simulations in a cloud environment," *Futur. Gener. Comput. Syst.*, vol. 115, pp. 497–516, 2021, doi: 10.1016/j.future.2020.09.016.

[27] H. A. Almashhadani, X. Deng, S. N. A. Latif, M. M. Ibrahim, and O. H. R. AL-hwaidi, "Deploying an efficient and reliable scheduling for mobile edge computing for IoT applications," *Mater. Today Proc.*, vol. 80, pp. 2850–2857, 2023, doi: 10.1016/j.matpr.2021.07.050.

[28] X. Wu, Y. Yang, Y. Sun, Y. Xie, X. Song, and B. Huang, "Dynamic regional splitting planning of remote sensing satellite swarm using parallel genetic PSO algorithm," *Acta Astronaut.*, vol. 204, no. April 2022, pp. 531–551, 2023, doi: 10.1016/j.actaastro.2022.09.020.

[29] Y. Lu, L. Liu, J. Gu, J. Panneerselvam, and B. Yuan, "EA-DFPSO: An intelligent energy-efficient scheduling algorithm for mobile edge networks," *Digit. Commun. Networks*, vol. 8, no. 3, pp. 237–246, 2022, doi: 10.1016/j.dcan.2021.09.011.

[30] A. B. Kanbar and K. Faraj, "Region aware dynamic task scheduling and resource virtualization for load balancing in IoT–fog multi-cloud environment," *Futur. Gener. Comput. Syst.*, vol. 137, pp. 70–86, 2022, doi: 10.1016/j.future.2022.06.005.

[31] Y. Zhang, Y. Liu, J. Zhou, J. Sun, and K. Li, "Slow-movement particle swarm optimization algorithms for scheduling security-critical tasks in resource-limited mobile edge computing," *Futur. Gener. Comput. Syst.*, vol. 112, pp. 148–161, 2020, doi: 10.1016/j.future.2020.05.025.

[32] N. C. Brintha, S. Benedict, and J. T. W. Jappes, "A bio-inspired hybrid computation for managing and scheduling virtual resources using cloud concepts," *Appl. Math. Inf. Sci.*, vol. 11, no. 2, pp. 565–572, 2017, doi: 10.18576/amis/110228.

[33] H. Zhao, N. Feng, F. Meng, Q. Wang, B. Wan, and J. Wang, "A Mapping-based Dynamic Semi-Online Task Scheduling Method for Minimizing Energy in Edge Computing," *2021 IEEE 23rd Int Conf High Perform. Comput. Commun. 7th Int Conf Data Sci. Syst. 19th Int Conf Smart City; 7th Int Conf Dependability Sensor, Cloud Big Data Syst. Appl.*, pp. 721–726, 2021, doi: 10.1109/HPCC-DSS-SmartCity-DependSys53884.2021.00118.

[34] J. Xu, Z. Hao, R. Zhang, and X. Sun, "A Method Based on the Combination of Laxity and Ant Colony System for Cloud-Fog Task Scheduling," *IEEE Access*, vol. 7, pp. 116218–116226, 2019, doi: 10.1109/ACCESS.2019.2936116.

[35] B. Hu, Y. Shi, and Z. Cao, "Adaptive Energy-Minimized Scheduling of Real-Time Applications in Vehicular Edge Computing," *IEEE Trans. Ind. Informatics*, vol. 19, no. 5, pp. 6895–6906, 2023, doi: 10.1109/TII.2022.3207754.

[36] J. Manivannan, "An Energy Efficient Scheduling in Industrial Internet of Things (IIoT)," *2023 Second Int. Conf. Adv. Comput. Intell. Commun.*, pp. 1–5, 2023, doi: 10.1109/ICACIC59454.2023.10435314.

[37] L. I. Tianze, W. U. Muqing, Z. Min, and L. Wenxing, "An Overhead-Optimizing Task Scheduling Strategy for Ad-hoc Based Mobile Edge Computing," pp. 5609–5622, 2017.

[38] X. Liu, "Computation Offloading and Task Scheduling with Fault-Tolerance for Minimizing Redundancy in Edge Computing," *2021 IEEE Int. Symp. Softw. Reliab. Eng. Work.*, pp. 198–209, 2021, doi: 10.1109/ISSREW53611.2021.00064.

[39] B. Sellami, A. Hakiri, S. Ben Yahia, and P. Berthou, "Deep Reinforcement Learning for Energy-Efficient Task Scheduling in SDN-based IoT Network", doi: 10.1109/NCA51143.2020.9306739.

[40] J. Bisht, "Energy Efficient and Optimized Makespan Workflow Scheduling Algorithm for Heterogeneous Resources in Fog-Cloud-Edge Collaboration," pp. 78–83, 2020, doi: 10.1109/WIECON-ECE52138.2020.9398042.

[41] M. Abdel-basset, R. Mohamed, M. Elhoseny, A. K. Bashir, A. Jolfaei, and N. Kumar, "Energy-Aware Marine Predators Algorithm for Task Scheduling in IoT-Based," vol. 17, no. 7, pp. 5068–5076, 2021, doi: 10.1109/TII.2020.3001067.

[42] M. Yuan and N. M. Freris, "Energy-aware online task dispatching and scheduling for edge systems with energy harvesting," *2022 13th Int. Conf. Netw. Futur.*, pp. 1–9, doi: 10.1109/NoF55974.2022.9942573.

[43] C. Wang, X. Yu, L. Xu, S. Member, and W. Wang, "Energy-Efficient Task Scheduling Based on Traffic Mapping in Heterogeneous Mobile-Edge Computing : A Green IoT Perspective," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 2, pp. 972–982, 2023, doi: 10.1109/TGCN.2022.3186314.

[44] P. Li, Z. Xiao, X. Wang, K. Huang, Y. Huang, and H. Gao, "EPtask : Deep Reinforcement Learning Based Energy-Efficient and Priority-Aware Task

Scheduling for Dynamic Vehicular Edge Computing," *IEEE Trans. Intell. Veh.*, vol. 9, no. 2, pp. 1830–1846, 2024, doi: 10.1109/TIV.2023.3321679.

[45] A. Satouf and A. Hamidoglu, "Grey Wolf Optimizer-based Task Scheduling for IoT-based Applications in the Edge Computing," *2023 Eighth Int. Conf. Fog Mob. Edge Comput.*, pp. 52–57, 2023, doi: 10.1109/FMEC59375.2023.10306148.

[46] X. Liu, "Low Energy Consumption and Time Deterministic Energy-saving Workflow Task Scheduling Algorithm based on DVFS," *2022 IEEE 9th Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/2022 IEEE 8th Int. Conf. Edge Comput. Scalable Cloud*, pp. 56–61, 2022, doi: 10.1109/CSCloud-EdgeCom54986.2022.00019.

[47] K. Li, "Scheduling Precedence Constrained Tasks for Mobile Applications in Fog Computing," *IEEE Trans. Serv. Comput.*, vol. 16, no. 3, pp. 2153–2164, 2023, doi: 10.1109/TSC.2022.3192095.

[48] S. Aleteemat and M. Shurman, "Task Offloading Scheduling in Fog Computing Using Hybrid Genetic Algorithm," *2023 14th Int. Conf. Inf. Commun. Syst.*, pp. 1–6, doi: 10.1109/ICICS60529.2023.10330532.

[49] Y. Luo, J. Wu, Y. Wu, and L. Chen, "Task Scheduling in Mobile Edge Computing with Stochastic Requests and M / M / 1 Servers," *2019 IEEE 21st Int. Conf. High Perform. Comput. Commun. IEEE 17th Int. Conf. Smart City; IEEE 5th Int. Conf. Data Sci. Syst.*, pp. 2379–2382, 2019, doi: 10.1109/HPCC/SmartCity/DSS.2019.00331.

[50] Z. Zhong, "Multiobjective African Vulture Scheduling Algorithm in Green Mobile Edge Computing," *2023 IEEE 14th Int. Conf. Softw. Eng. Serv. Sci.*, pp. 65–70, doi: 10.1109/ICSESS58500.2023.10293101.

[51] Q. Yan, L. Ma, and J. Sun, "Novel Bat Algorithms for Scheduling Independent Tasks in Collaborative Internet-of-Things," 2020, doi: 10.1109/HPCC-SmartCity-DSS50907.2020.00087.

[52] T. State and S. Networks, "Poster : A Dynamic Task Scheduling using Multi-Platoon Architecture in Vehicular Networks," *2022 IEEE 42nd Int. Conf. Distrib. Comput. Syst.*, pp. 1284–1285, 2022, doi: 10.1109/ICDCS54860.2022.00135.

[53] W. Zhu, X. Chen, L. Jiao, G. Min, and Y. Wang, "Priority Scheduling Strategy for Reduced Energy Consumption in UAV-aided 6G Green Mobile Edge Computing," *2023 IEEE Int. Conf. Syst. Man, Cybern.*, pp. 44–50, 2023, doi: 10.1109/SMC53992.2023.10394133.

[54] X. Zhao, X. Guo, Y. Zhang, and W. Li, "A Parallel-batch Multi-objective Job Scheduling Algorithm in Edge Computing," *2018 IEEE Int. Conf. Internet Things IEEE Green Comput. Commun. IEEE Cyber, Phys. Soc. Comput. IEEE Smart Data*, pp. 510–516.

[55] P. Kaur, "Applying a cost effective hybrid approach to task scheduling and matching for computational grids," 2017.

[56] M. Abdel-Basset, D. El-Shahat, K. Deb, and M. Abouhawwash, "Energy-aware whale optimization algorithm for real-time task scheduling in multiprocessor systems," *Appl. Soft Comput. J.*, vol. 93, p. 106349, 2020, doi: 10.1016/j.asoc.2020.106349.

[57] Y. Lu, L. Liu, J. Gu, J. Panneerselvam, and B. Yuan, "EA-DFPSO : An intelligent energy-ef fi cient scheduling algorithm for mobile edge networks," *Digit. Commun. Networks*, vol. 8, no. 3, pp. 237–246, 2022, doi: 10.1016/j.dcan.2021.09.011.

[58] B. M. Sahoo, T. Amgoth, and H. M. Pandey, "Particle swarm optimization based energy efficient clustering and sink mobility in heterogeneous wireless sensor network," *Ad Hoc Networks*, vol. 106, p. 102237, 2020, doi: 10.1016/j.adhoc.2020.102237.

[59] A. Verma and S. Kaushal, "A hybrid multi-objective Particle Swarm Optimization for scientific workflow scheduling," *Parallel Comput.*, vol. 62, pp. 1–19, 2017, doi: 10.1016/j.parco.2017.01.002.

[60] C. Computing, "Performance Optimization of Task Scheduling in Fog and Edge Computing using meta-heuristic algorithms for IoT Networks : A Comparative Study Pratyush Nigel Baxla National College of Ireland Supervisor :".

[61] N. Binti and M. Isa, "Smart Grid Technology : Communications , Power Electronics and Control System," *2015 Int. Conf. Sustain. Energy Eng. Appl.*, pp. 10–14, 2015, doi: 10.1109/ICSEEA.2015.7380737.

[62] H. Algorithm, "A Quality-of-Service-Aware Service Composition Method in the," pp. 1–29, 2023.

[63] M. Hachicha, R. Ben Halima, and A. H. Kacem, "ScienceDirect Formal Formal Verification Verification approaches approaches of of Self-adaptive Self-adaptive Systems : Systems : A A Survey," *Procedia Comput. Sci.*, vol. 159, pp. 1853–1862, 2019, doi: 10.1016/j.procs.2019.09.357.

[64] W. Shu and Y. Li, "Joint of fl oading strategy based on quantum particle swarm optimization for MEC-enabled vehicular networks," *Digit. Commun. Networks*, vol. 9, no. 1, pp. 56–66, 2023, doi: 10.1016/j.dcan.2022.03.009.

[65] R. Poli, J. Kennedy, and T. Blackwell, "Quantification & Assessment of the chemical form of residual gadolinium in the brain.pdf," *Swarm Intell*, vol. 1, pp. 33–57, 2007, doi: 10.1007/s11721-007-0002-0.

[66] S. Pervaiz *et al.*, "Comparative Research Directions of Population Initialization Techniques using PSO Algorithm," *Intell. Autom. Soft Comput.*, vol. 32, no. 3, pp. 1427–1444, 2022, doi: 10.32604/IASC.2022.017304.

[67] Y. Zhang, S. Wang, and G. Ji, "A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications," vol. 2015, 2015, doi: 10.1155/2015/931256.

[68] Q. You and B. Tang, "Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial internet of things," 2021.

[69] J. Liu, C. Li, and Y. Luo, "Efficient resource allocation for IoT applications in mobile edge computing via dynamic request scheduling optimization," *Expert Syst. Appl.*, vol. 255, no. PC, p. 124716, 2024, doi: 10.1016/j.eswa.2024.124716.

[70] M. Saad, R. N. Enam, and R. Qureshi, "Optimizing multi-objective task scheduling in fog computing with GA-PSO algorithm for big data application," *Front. Big Data*, vol. 7, 2024, doi: 10.3389/fdata.2024.1358486.

[71] C. Huang, H. Wang, L. Zeng, and T. Li, "Resource Scheduling and Energy Consumption Optimization Based on Lyapunov Optimization in Fog Computing," *Sensors*, vol. 22, no. 9, 2022, doi: 10.3390/s22093527.

[72] W. Zhang, H. Xie, B. Cao, and A. M. K. Cheng, "Energy-Aware Real-Time Task Scheduling for Heterogeneous Multiprocessors with Particle Swarm Optimization Algorithm," *Math. Probl. Eng.*, vol. 2014, 2014, doi: 10.1155/2014/287475.

[73] B. Wang, J. Cheng, J. Cao, C. Wang, and W. Huang, "Integer particle swarm optimization based task scheduling for device-edge- cloud cooperative computing to improve SLA satisfaction," vol. 1, pp. 1–22, 2023, doi: 10.7717/peerj-cs.893.

[74] M. Zhang, L. Liu, C. Li, H. Wang, and M. Li, "A Particle Swarm Optimization Method for AI Stream Scheduling in Edge Environments," *Symmetry (Basel).*, vol. 14, no. 12, pp. 1–18, 2022, doi: 10.3390/sym14122565.

[75] Y. Xie, Y. Zhu, Y. Wang, Y. Cheng, and R. Xu, "A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud – edge environment," *Futur. Gener. Comput. Syst.*, vol. 97, pp. 361–378, 2019, doi: 10.1016/j.future.2019.03.005.

[76] S. Bansal and H. Aggarwal, "A Hybrid Particle Whale Optimization Algorithm with application to workflow scheduling in cloud–fog environment," *Decis. Anal. J.*, vol. 9, no. October, p. 100361, 2023, doi: 10.1016/j.dajour.2023.100361.

[77] N. Arora and R. K. Banyal, "Workflow scheduling using particle swarm optimization and gray wolf optimization algorithm in cloud computing," *Concurr. Comput. Pract. Exp.*, vol. 33, no. 16, pp. 1–16, 2021, doi: 10.1002/cpe.6281.

[78] G. shun Yao, Y. sheng Ding, and K. rong Hao, "Multi-objective workflow scheduling in cloud system based on cooperative multi-swarm optimization algorithm," *J. Cent. South Univ.*, vol. 24, no. 5, pp. 1050–1062, 2017, doi: 10.1007/s11771-017-3508-7.

[79] I. Casas, J. Taheri, R. Ranjan, and A. Y. Zomaya, "PSO-DS: a scheduling engine for scientific workflow managers," *J. Supercomput.*, vol. 73, no. 9, pp. 3924–3947, 2017, doi: 10.1007/s11227-017-1992-z.

[80] X. Ren, Z. Zhang, S. Chen, and K. Abnoosian, "An energy-aware method for task allocation in the Internet of things using a hybrid optimization algorithm," *Concurr. Comput. Pract. Exp.*, vol. 33, no. 6, pp. 1–14, 2021, doi: 10.1002/cpe.5967.

[81] W. Xiong, B. Guo, and S. Yan, "Energy consumption optimization of processor scheduling for real-time embedded systems under the constraints of sequential relationship and reliability," *Alexandria Eng. J.*, vol. 61, no. 1, pp. 73–80, 2022, doi: 10.1016/j.aej.2021.04.071.

[82] M. R. Jose and S. M. C. Vigila, "F-CAPSO: Fuzzy chaos adaptive particle swarm optimization for energy-efficient and secure data transmission in MANET," *Expert Syst. Appl.*, vol. 234, no. January, p. 120944, 2023, doi: 10.1016/j.eswa.2023.120944.

[83] H. Hafsi, H. Gharsellaoui, and S. Bouamama, "Genetically-modified Multi-objective Particle Swarm Optimization approach for high-performance computing workflow scheduling," *Appl. Soft Comput.*, vol. 122, p. 108791, 2022, doi: 10.1016/j.asoc.2022.108791.

[84] J. Bi, H. Yuan, and S. Duanmu, "Energy-efficient Task Offloading Using Hybrid Particle Swarm Optimization with Genetic Operations in Smart Edge," *IFAC-PapersOnLine*, vol. 53, no. 5, pp. 19–24, 2020, doi: 10.1016/j.ifacol.2021.04.122.

[85] N. Mansouri, B. Mohammad Hasani Zade, and M. M. Javidi, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory," *Comput. Ind. Eng.*, vol. 130, no. March, pp. 597–633, 2019, doi: 10.1016/j.cie.2019.03.006.

[86] S. S. Gill, R. Buyya, I. Chana, M. Singh, and A.

Abraham, "BULLET: Particle Swarm Optimization Based Scheduling Technique for Provisioned Cloud Resources," *J. Netw. Syst. Manag.*, vol. 26, no. 2, pp. 361–400, 2018, doi: 10.1007/s10922-017-9419-y.

[87] H. P. Hsu and C. N. Wang, "Hybridizing Whale Optimization Algorithm With Particle Swarm Optimization for Scheduling a Dual-Command Storage/Retrieval Machine," *IEEE Access*, vol. 11, no. January, pp. 21264–21282, 2023, doi: 10.1109/ACCESS.2023.3246518.

[88] A. Verma and S. Kaushal, "Bi-Criteria Priority based Particle Swarm Optimization workflow scheduling algorithm for cloud," *2014 Recent Adv. Eng. Comput. Sci. RAECS 2014*, no. May, 2014, doi: 10.1109/RAECS.2014.6799614.

[89] Z. Zhan, X. Liu, Y. Gong, and J. U. N. Zhang, "Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches," vol. 47, no. 4, 2015.

[90] N. G. Puttaswamy, A. N. Murthy, and H. Degha, "A Comparative Review of Internet of Things Model Workload Distribution Techniques in Fog Computing Networks," pp. 21–46, 2024.

[91] S. M. R. Dibaj, "A cloud dynamic online double auction mechanism (DODAM) for sustainable pricing A cloud dynamic online double auction mechanism (DODAM) for," *Telecommun. Syst.*, no. December, 2020, doi: 10.1007/s11235-020-00688-4.

[92] T. Lu, F. Zeng, G. Chen, W. Shu, J. Shen, and W. Zhang, "A Novel Hybrid Model for Task Dependent Scheduling in Container-based Edge Computing," *2021 IEEE Int. Conf. Commun. Work. ICC Work. 2021 - Proc.*, 2021, doi: 10.1109/ICCWorkshops50388.2021.9473877.

[93] F. A. Omara and M. M. Arafa, "Genetic algorithms for task scheduling problem," *J. Parallel Distrib. Comput.*, vol. 70, no. 1, pp. 13–22, 2010, doi: 10.1016/j.jpdc.2009.09.009.

[94] H. Hussain *et al.*, "Energy Efficient Real-Time Tasks Scheduling on High-Performance Edge-Computing Systems Using Genetic Algorithm," *IEEE Access*, vol. 12, no. April, pp. 54879–54892, 2024, doi: 10.1109/ACCESS.2024.3388837.

[95] J. He, "Optimization of Edge Delay Sensitive Task Scheduling Based on Genetic Algorithm," *Proc. - 2022 Int. Conf. Algorithms, Data Mining, Inf. Technol. Admit 2022*, pp. 155–159, 2022, doi: 10.1109/ADMIT57209.2022.00032.

[96] K. Shao, Y. Song, and B. Wang, "PGA: A New Hybrid PSO and GA Method for Task Scheduling with Deadline Constraints in Distributed Computing," *Mathematics*, vol. 11, no. 6, pp. 1–16, 2023, doi: 10.3390/math11061548.

[97] M. Akbari, H. Rashidi, and S. H. Alizadeh, "An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems," *Eng. Appl. Artif. Intell.*, vol. 61, no. March, pp. 35–46, 2017, doi: 10.1016/j.engappai.2017.02.013.

[98] A. A. Al-Habob, O. A. Dobre, and A. Garcia Armada, "Sequential task scheduling for mobile edge computing using genetic algorithm," *2019 IEEE Globecom Work. GC Wkshps 2019 - Proc.*, 2019, doi: 10.1109/GCWkshps45667.2019.9024374.

[99] A. Mahmood, S. A. Khan, F. Albalooshi, and N. Awwad, "Energy-aware real-time task scheduling in multiprocessor systems using a hybrid genetic algorithm," *Electron.*, vol. 6, no. 2, 2017, doi: 10.3390/electronics6020040.

[100] S. G. Ahmad, E. U. Munir, and W. Nisar, "PEGA: A Performance Effective Genetic Algorithm for task scheduling in heterogeneous systems," *Proc. 14th IEEE Int. Conf. High Perform. Comput. Commun. HPCC-2012 - 9th IEEE Int. Conf. Embed. Softw. Syst. ICESS-2012*, no. June, pp. 1082–1087, 2012, doi: 10.1109/HPCC.2012.158.

[101] F. Hoseiny, S. Azizi, M. Shojafar, F. Ahmadiazar, and R. Tafazolli, "PGA: A priority-aware genetic algorithm for task scheduling in heterogeneous fog-cloud computing," *IEEE INFOCOM 2021 - IEEE Conf. Comput. Commun. Work. INFOCOM WKSHPS 2021*, no. June, 2021, doi: 10.1109/INFOCOMWKSHPS51825.2021.9484436.

[102] A. A. Al-Habob, O. A. Dobre, A. G. Armada, and S. Muhaidat, "Task scheduling for mobile edge computing using genetic algorithm and conflict graphs," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8805–8819, 2020, doi: 10.1109/TVT.2020.2995146.

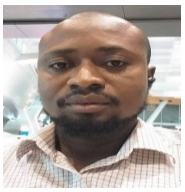
[103] Y. Xu, K. Li, J. Hu, and K. Li, "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues," *Inf. Sci. (Ny).*, vol. 270, pp. 255–287, 2014, doi: 10.1016/j.ins.2014.02.122.

[104] M. Dorigo, "The Ant Colony Optimization Metaheuristic : Algorithms , Applications , and Advances ∗ Technical Report IRIDIA-2000-32," no. June, 2014.

[105] Y. Nan, "An improved ant colony optimization algorithm based on immunization strategy," *Adv. Mater. Res.*, vol. 490–495, pp. 66–70, 2012, doi: 10.4028/www.scientific.net/AMR.490-495.66.

[106] M. Farid, R. Latip, M. Hussin, N. Asilah, and W. Abdul, "SS symmetry A Survey on QoS Requirements Based on Particle Swarm Optimization Scheduling Techniques for

Workflow Scheduling in Cloud Computing," 2020.
[107] W. N. Chen, Y. Shi, and J. Zhang, "An ant colony optimization algorithm for the time-varying workflow scheduling problem in grids," *2009 IEEE Congr. Evol. Comput. CEC 2009*, pp. 875–880, 2009, doi: 10.1109/CEC.2009.4983037.
[108] E. Pacini, C. Mateos, and C. García Garino, "Balancing throughput and response time in online scientific Clouds via Ant Colony Optimization (SP2013/2013/00006)," *Adv. Eng. Softw.*, vol. 84, pp. 31–47, 2015, doi: 10.1016/j.advengsoft.2015.01.005.
[109] M. P. Mathiyalagan, S. Suriya, and S. N. Sivanandam, "Modified Ant Colony Algorithm for Grid Scheduling," *IJCSE) Int. J. Comput. Sci. Eng.*, vol. 02, no. 02, pp. 132–139, 2010.
[110] A. Liu and Z. Wang, "Grid task scheduling based on adaptive ant colony algorithm," *Proc. - Int. Conf. Manag. e-Commerce e-Government, ICMeCG 2008*, pp. 415–418, 2008, doi: 10.1109/ICMECG.2008.50.
[111] B. Chandra Mohan and R. Baskaran, "A survey: Ant Colony Optimization based recent research and implementation on several engineering domain," *Expert Syst. Appl.*, vol. 39, no. 4, pp. 4618–4627, 2012, doi: 10.1016/j.eswa.2011.09.076.
[112] A. V. Baterina and C. Oppus, "Image edge detection using ant colony optimization," *WSEAS Trans. Signal Process.*, vol. 6, no. 2, pp. 58–67, 2010, doi: 10.5958/2455-7110.2017.00025.8.
[113] M. Mavrovouniotis, F. M. Muller, and S. Yang, "Ant Colony Optimization with Local Search for Dynamic Traveling Salesman Problems," *IEEE Trans. Cybern.*, vol. 47, no. 7, pp. 1743–1756, 2017, doi: 10.1109/TCYB.2016.2556742.
[114] A. Gonzalez-Pardo, J. Del Ser, and D. Camacho, "Comparative study of pheromone control heuristics in ACO algorithms for solving RCPSP problems," *Appl. Soft Comput. J.*, vol. 60, pp. 241–255, 2017, doi: 10.1016/j.asoc.2017.06.042.
[115] E. J. Chang, H. K. Hsin, C. H. Chao, S. Y. Lin, and A. Y. A. Wu, "Regional ACO-based cascaded adaptive routing for traffic balancing in mesh-based network-on-chip systems," *IEEE Trans. Comput.*, vol. 64, no. 3, pp. 868–875, 2015, doi: 10.1109/TC.2013.2296032.
[116] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous vehicular edge computing framework with ACO-based scheduling," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10660–10675, 2017, doi: 10.1109/TVT.2017.2714704.
[117] X. Yu, W. N. Chen, T. Gu, H. Yuan, H. Zhang, and J. Zhang, "ACO-A∗: Ant Colony Optimization plus A∗ for 3-D Traveling in Environments with Dense Obstacles," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 617–631, 2019, doi: 10.1109/TEVC.2018.2878221.



Jafar Aminu earned his bachelor's degree in computer science from Kebbi State University of Science and Technology, Aleiro, in 2012. He completed his master's degree in computer science in 2021 and is currently pursuing a Ph.D. at the Faculty of Computer Science and Information Technology. His research interests encompass computer networks, workflow and task offloading, edge computing, the Internet of Things, optimization, workload allocation, and task offloading.

Rohaya Latip (Member, IEEE) earned her bachelor's degree in computer science from Universiti Teknologi Malaysia in 1999. She went on to complete her M.Sc. in distributed systems and her Ph.D. in distributed databases at Universiti Putra Malaysia. From 2011 to 2012, she served as the Head of the HPC Section at Universiti Putra Malaysia and has been involved in consulting projects such as the Campus Grid Project and the Wireless for Hostel in Campus UPM Project. Currently, she is an Associate Professor in the Faculty of Computer Science and Information Technology at Universiti Putra Malaysia. Additionally, she heads the Department of Communication Technology and Networks and is a Co-Researcher at the Institute for Mathematical Research (INSPEM). Her research interests include big data, cloud and grid computing, network management, and distributed databases.

Zurina Mohd Hanapi (Member, IEEE) earned her B.Sc. in Computer and Electronic Systems from the University of Strathclyde, Glasgow, U.K., in 1999, her M.Sc. in Computer and Communication System Engineering from Universiti Putra Malaysia (UPM), Malaysia, in 2004, and her Ph.D. from Universiti Kebangsaan Malaysia in 2011. She is currently an Associate Professor in the Department of Communication Technology and Networks at the Faculty of Computer Science and Information Technology, UPM, where she has been a lecturer since 2004. She has

published over 70 papers in prominent journals and conferences focusing on security and wireless sensor networks. Her research interests include security, routing, wireless sensor networks, wireless networks, distributed computing, and cyber-physical systems. She is also a member of the Malaysian Security Committee Research.

Kamarudin Shafinah earned her Diploma, Bachelor's, and M.Sc. degrees in Computer Science from Universiti Putra Malaysia in 2000, 2003, and 2009, respectively, and completed her Ph.D. at Universiti Kebangsaan Malaysia in 2016. She is currently a Senior Lecturer at Universiti Putra Malaysia, where she has been actively involved in authoring book chapters and publishing a wide range of papers in both journals and conferences. Her research focuses on computer networks, management information systems, and related areas.CT for agriculture, and scholarship for teaching and learning (SoTL).

Dr. Danlami Gabi is a Senior Lecturer in the Department of Computer Science at Kebbi State University of Science and Technology, Aliero, Nigeria. He earned his Ph.D. from Universiti Teknologi Malaysia between 2015 and 2018 and completed a Postdoctoral Fellowship in Computing Science at Umea University, Sweden, from 2018 to 2021. He holds a Master’s degree in Information Security and Computer Forensics from the University of East London, UK, which he received in 2012. His research interests encompass Swarm Intelligence, Scheduling, Cloud and Edge Computing, Game Theory, Network Security, and Machine Learning.