

***University of Ibadan Journal of
Science and Logics in ICT
Research (UIJSLICTR)***
ISSN: 2714-3627

A Journal of the Faculty of Computing, University of Ibadan, Ibadan, Nigeria

Volume 17 No. 1, June 2026

***journals.ui.edu.ng/uijslictr
http://uijslictr.org.ng/
uijslictr@gmail.com***



A Criticality-aware Task Scheduling and Offloading Optimization Model for Mobile Edge-Cloud-bound Tasks: A Review

¹Babalola O. O. and ²Oguntunde T.

^{1,2} Department of Information and Communication Technology and Cyber Security,
University of Ibadan

Emails: dlensplc@gmail.com , t.oguntunde@ui.edu.ng

Abstract

There have been innovative surveys and researches on efficient scheduling of computationally intensive and latency sensitive mobile tasks from smart devices onto cloud and edge environments. These surveys are geared towards improving the performance of these tasks. However a very insignificant attention has been paid to the review of criticality-based separation which exists among the latency sensitive tasks; some latency sensitive are non-critical (LSNCTs) while some are critical (LSCTs). This trivial attention is insensitive as it could consequently constitute an undue increase in service times of the LSCTs, lead to critical application failure and possibly loss of lives. Therefore, this paper presents a review on A Criticality-aware Task Scheduling and Offloading Optimization Model for Mobile Edge-cloud Bound Tasks. The review leveraged 69 papers and 42 of them are the related works. Eighteen (18) out of the 42 reviewed studies are from 2022 to 2026, the related papers' years spanned through 2002 to 2026 with Journal articles having 90% of the reviewed papers and mostly from IEEE Xplore database. This statistics is significant and of benefit as it establishes the historical depth, reliability, academic freshness and contemporariness in the reviewed papers. The result of the paper reveals the various drawbacks associated with the existing literatures to give room for future works. In addition, it gives an insight to the advancement in the body of knowledge especially in the area of prioritizing LSCTs emanating from the smart devices such as mobile devices, Internet of Things (IoT) and Unmanned Aerial Vehicles (UAV).

Keywords: Latency Sensitive Critical Task, Latency Sensitive Non-Critical Tasks, Internet of Things, Unmanned Aerial Vehicle, Resource Allocation and Task Scheduling

1.0 Introduction

The advent of Internet of Things (IoTs) [1] and sophistication in Smart Mobile Devices (SMDs) [2] has unaguably promoted the development of various mobile applications. Most of these apps such as remote surgery, virtual and augmented reality, face and gesture recognition, online interactive gaming are resource demanding and latency sensitive. The SMDs such as the wearable, IoT, mobile devices and Unmanned Aerial Vehincles (UAVs) [3] are however limited in their compute power, storage and battery strength. Undue delay and application failure are few of the inherent problems due to

this gap. The solution to this gap between the resource constraint devices and the computational intensive apps is computational offloading of these tasks into a resource-rich environment called cloud. According to the National Institute of Standards and Technology (2008),

Cloud computing is defined as “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [4]. The Mobile Cloud Computing (MCC) Forum considers MCC as “an infrastructure where both the data storage and the data processing happen outside of the mobile device [5].

Babalola O. O. and Oguntunde T. (2026). A Criticality-aware Task Scheduling and Offloading Optimization Model for Mobile Edge-Cloud-bound Tasks: A Review. *University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR)*, Vol. 17 No. 1, pp. 166 – 183

Despite the cloud's unlimited and ubiquitous shareable services [6], leveraging the cloud resource rich environment usually incurs some delays due to mobile device-Cloud long distance [7]. In addition, Communication latency are inherent problems associated with Cloud computing offloading. Therefore, sending a latency sensitive but non-computationally intensive mobile apps to the cloud adds to the overall service times of such applications because the applications suffer undue increased latency. A cloud computing extended paradigm, which replicates the cloud's services [8] in a close vicinity to thin clients such as mobile devices [9] with minimal latency, has been proposed by many researchers. This is called Mobile Edge Computing (MEC). MEC helps IoT mobile apps such as intelligent buildings and vehicles, and smart grids run in a timely manner [10].

MEC offers reduced service delivery latency, [11] minimized energy consumption and agile computational services. However, due to the sizes of their devices, [12] [13] MEC is resource constrained. MEC cannot therefore, replace Cloud environment. Highly computationally intensive tasks would still require an offloading to cloud data center to minimize computational cost. With this, to solely rely on offloading tasks to edge would be inefficient. Therefore, leveraging MEC and Cloud environment is always advisable using Open Fog Consortium Architecture (OFC) [14].

Efficient use of cloud and edge resources and arranging the offloaded tasks require appropriate mapping [15]. The solution to this mapping is scheduling. The scheduling of tasks in the edge and cloud environment involves allocating the tasks in an efficient manner such that certain performance metrics such as throughput, response and turnaround times, resource utilization and waiting time are optimized [16]. Scheduling is therefore geared towards improving and enhancing some performance metrics, a process called optimization.

Mobile tasks are heterogeneous and therefore usually have different processing requirements and a resource rich environment guarantees quality performance. [17]. Notable among the use cases leveraging the rich environment are social network, mailing apps, news app, Augmented Reality (AR) navigation apps,

content delivery apps for streaming, online gaming, disaster management apps, video conferencing, avionics, telemedicine app, time critical IoT and autonomous Driver, to mention but a few. Among these tasks, by their type, it is evident that applications are not created equally. These tasks have different computation intensiveness, business values, economic importance and above all criticality level to human lives, user experience and property.

As they are being generated from their various smart devices (Mobile phone, IoT, UAVs) to be scheduled at the resource rich-environment, they therefore require different processing requirements, latency requirement and urgency. Deadline is a very important factor during scheduling of tasks. It is defined as the time frame usually specified by users within which a task must be completed without leading to application failure, loss of business value, loss of properties and even danger to lives [18]. It could determine the criticality and the urgency of a task. Meeting up with the deadline of a task is always the utmost consideration, although, most scheduling frameworks are silent on this crucial and lifesaving consideration. The existing scheduling frameworks have focused on the optimization of some of the performance metrics such as resource utilization, waiting time [19][20] turnaround time, energy consumption but silent on giving cognizance to the fact that some tasks are more urgent than others.

Latency, defined as the time interval or delay between when a mobile task is generated by its originating smart device and the time it is done executing and sent back to this device. Latency is a huge factor to determine if the deadline of a task will be met or not. Of course many frameworks have been put forward to provide guarantees that tasks are successfully executed and finished before their deadlines through minimization of latency using different heuristics and meta-heuristics scheduling techniques. However, being latent-unfriendly or delay sensitive and being of low deadline does not always translate to being critical and urgent [21].

In other words, an application may be delay sensitive but forgiving while some may be latent-unfriendly and at the same time critical and non-forgiving. The latter are the categories of task that if their deadlines are not met can lead

to loss of properties even loss of lives [22]. This thin line has not been identified in the existing reviews, tasks were being generically scheduled based on their deadline and latency values without recognizing that deadline values do not translate to criticality. This generic technique of equally giving attention to all latency sensitive jobs endangers fast execution of the latency sensitive critical tasks (LSNTs) such as mission and safety critical applications e.g. avionics, aerospace, nuclear plants, remote surgery, industrial automation.

The existing reviews failed to consider that some tasks are more urgent than the others and they require different processing power or resources even among applications of the same type. For instance, telemedicine tasks that are usually considered critical, even among them, some are more critical than the others, the existing researches did not give cognizance to this. The consequence of this tradition is that the latent-unfriendly critical tasks suffer undue and longer latency/ turnaround time, lower throughput and lower order of precedence when compared with a scheduling whose major priority determiner is the level of urgency and criticality.

In addition, the average-based performance metrics values [23] of the existing works are insufficient to cater for these critical mission tasks; they are still sub-optimal to meet up with the permissible delay of these tasks, hence, a demand for more robust solutions to optimization problem is desirable [24]. Therefore, this article is aimed at giving an overview on the different research advances in dispatching, classifying and prioritizing different tasks through scheduling with appropriate requirements towards reducing the latency, increasing the throughput in edge, edge consortium and cloud computing.

1.1 The Contribution and the Scope of this Paper

The existing research review on critical application related scheduling and resource allocations gave the account of works published between 2018 and 2023 [25]. This is a limited time frames.

In contrast to this, this review article brought an additional and broader spectrum to what have been done on mission-critical application scheduling. This includes

1. Historical depth advantage: This paper reviewed the earlier years from 2002 to the most recent works in 2025.
2. The recent advanced technology in processors and memory of smart devices significantly justifies that some non-latent friendly, critical mobile tasks with low processing requirement could be executed locally without necessarily sending them to environment like cloud. The advantage with this advancement are zero communication latency, load balancing and the urgent tasks which are not necessarily resource constrained are attended to in a timely manner. As a result of this benefits, this article further considered frameworks that factored in running tasks at the smart device layer
3. The current review paper on criticality related schedulers ignored offloading and execution-layer categorization. This means that the review missed one of the most important levers for managing latency in mission-critical scenarios. This review paper is therefore of utmost importance to supporting future as it considered other relevant papers which were not considered in the existing review paper. This helps accuracy and to avoid misleading the entire research community.

2. The Need for this Review

As earlier mentioned, the existing framework [25] did a systematic literature review on resource management for mission-critical applications at the edge. They engaged a reliable methodology and looked at the metrics that are matter for crucial workloads in edge environments. Yet, the work did not consider works that recognize the effect of transmission latency vis-a-vis the location where a particular task is run. Depending on location, WLAN, MAN or/and WAN latency are possible, the work did had a poor offloading technique and there's no categorization by execution layer. That feels like a major gap because in edge computing, the whole game is deciding where to run a task, and offloading decision directly determines tasks' latency profile. This review paper is therefore of utmost importance to supporting the future as it considered other relevant papers which were not considered in the existing researches for accuracy and to avoid misleading the entire research community.

3.0 The Thematic Organization

The 42 reviewed works whose aims were to improve the performance of latency sensitive tasks factored in the concepts of execution layer, task allocation and scheduling and resource utilization differently. Therefore, the reviews would be discussed based on the below categorizations:

3.1 Tasks Execution Layer

In the quest of achieving an optimal service offloading scheme and efficient service offloading in edge environment, Samanta and Chang [26] proposed an adaptive service offloading for revenue maximization in mobile edge computing with delay-constraint. In their works, two different natures of workloads were considered, the delay tolerant and delay sensitive tasks whereby the later were considered edge bound tasks while the former were declared cloud candidates during executions. They managed to achieve higher resource utilization and minimized total service time but the work did not cater for minimization of latency for the individual tasks that are latency sensitive. Even, among the latency sensitive tasks, the work failed to identify the critical ones.

Atan *et. al.* [27] posited an AI-based strategy to ensure that the latency sensitive tasks sent by Internet of Things devices onto edge are run with minimised service time by the edge servers. The work considered two execution environments for tasks; the local and edge. In the policy, local execution was allowed if the sending smart device has enough capacity to run tasks. Secondly, if the processing requirement of a task is more than that of the local device, it is sent to the edge. However, if none of the execution layers can sufficiently run the task, the task is dropped. This is a big setback in the part of the framework as it amounts to application failure.

A task offloading optimization model which considered a three-tier layer for running tasks was introduced by Naouri [28]. The approach includes cloud, edge and cloudlet. It is a greedy policy which scheduled tasks onto devices based on their compute powers was employed. Communication cost was mitigated in this model by optimizing task offloading decision. A major setback in this model is that offloading decision was based on computational intensiveness and latency. Another weakness is that the urgency of task was not a determining indicator.

A framework that encompasses two main levels: an end-device level and an edge level was established by Zhang [29]. In this approach, resource intensive jobs are intelligently moved to the best location in compliance with the hard constraints on latency. In order to prevent overburdening of devices, processing power is distributed among user devices on the terminal level. The originating device layer is composed of the client devices which generated some computationally intensive tasks that could not be executed by the devices. At the edge and cloud environments, Ant Colony Optimization (ACO) scheduling policy was deployed. It utilized the finest and the gentlest scheduling strategy. The policy helps to minimize battery utilization and service time of all tasks. The importance of balancing costs of the end-user with revenues for the service provider in deployment of a various tasks in proximity to network infrastructure was also emphasized in the study. Priority score was built at a very high level based mainly on deadlines. Unfortunately, tasks deadline values does not necessarily translate to urgency.

Li and Zhang [30] proposed Delay Strategy for Service Cache and Task Offloading in Three-Tier Architecture Mobile Edge Computing System. Li and Zhang [30] Introduced “Delay Control Strategy Joint Service Caching and Task Offloading (DCS-OCTO)” for mobile edge computing. They leaned on Lyapunov optimization and Gibbs sampling to keep things in check. Simulations showed a reduction in latency while keeping battery use reasonably low. Nevertheless, their work failed to make provision for prioritizing urgent tasks among the tasks being scheduled.

Lin *et. al.* [31] suggested Energy and performance-aware task scheduling in a mobile cloud computing environment. They proposed an algorithm that starts by scheduling for minimal delay and considered offloading tasks to either cloud or run tasks locally. The Major contributions include improved time management, resource matching and fault tolerance. Nevertheless, the strategy would have been more effective if edge environment was leveraged, this would have invariably given room for more minimal delay.

Concurrent IoT applications placement which leveraged memetic algorithm was put forward

by Goudarzi *et. al.* [32]. The study contributed by minimizing the average time of tasks' execution and energy consumption of mobile applications emanating from IoT devices, fog and edge devices. The strategy however did not identify the relevance of criticality among IoT applications thus scheduling and minimizing their execution time without preference.

Multi-user mobile computation offloading and transmission scheduling mechanism MOTM was proposed by Yi *et. al.* [33] to jointly determine the computation offloading scheme, the transmission scheduling discipline and the pricing rule. When a task is generated, it could either be locally run or sent to the edge through an uplink transmission. They considered the random arrivals of heterogeneous latency-sensitive computation tasks before offloading. Edge computing environments are however inherently characterized with average low compute power resources; delay-sensitive tasks that are resource-intensive tasks will therefore suffer since the compute rich environment was not considered.

3.2 Resource Allocation Determiner

The latency, the throughput and service time performance that a smart mobile task would witness depend, among other factors, the transmission medium quality [34], the task's requirements and the offloading techniques used especially for the Internet of Things applications. Of course, the decentralization of services by cloud in close proximity to devices helps minimizing the transmission latency but the computational latency incurred at the edge due to the associated physical limitations of the network devices is still significant. This was a rationale behind the fuzzy logic approach which was proposed by Almutairi and Aldossary [35].

Resource utilization, heterogeneity and task characteristics such as, latency sensitivity, processing requirement and bandwidth were used to decide the exact location to run a particular task in this approach. On top of these characteristics (task lengths, network demand delay sensitivity and the virtual machine utilizations), the researchers came up with a 4-input fuzzy rules-based table which accommodates about sixteen possible application nature and the corresponding location of running each of them. The work did not only establish the impact of offloading

decisions (a function of the computation and communication effects and location of execution), on the service time of latency-sensitive tasks but also contributed to minimization of task response times in edge-cloud consortium. Yet, as all-encompassing as the model appears, delay sensitivity was considered as deciding characteristic for resource allocation, this does not keenly recognize the criticality of some tasks among others. In addition, computationally intensiveness of tasks were also considered during task allocation which also had an issue of poor resource utilization

According to Lakhan *et. al.* [36] delay sensitive task that has the least deadline should be given more attention and should be mapped to resources with the best compute power when compared with the delay tolerant tasks. This was achieved by formulating Joint Task Offloading and Scheduling (JTOS) framework. With this framework, Internet of Things' tasks had 39% reduction in computational cost and 35% reduction in transmission latency when benchmarked against the previous works. The delay minimization was however for all applications; no cognizance was given to the fact that some applications are more delay sensitive and critical than the others. There was no cognizance to the fundamental that some tasks are delay sensitive but not critical. Some low deadline apps are not critical, this fundamental was not recognized. Therefore, low deadline tasks that are critical (Hard real time) suffer delay.

In the work by Iqbal *et. al.* [37], priority-based round robin algorithm with variable time slice was considered to have a maximized resource utilization and minimized task completion time using enhanced Round Robin scheduling algorithm variant incorporating dynamic time quantum and priority. Tasks were sorted based on shortest job first and arranged with the shortest job having the highest priority. Although, the work claimed to be relevant in real time systems but it is more of OS-based. The work also claimed to minimize the computational latency of urgent tasks but contrariwise, tasks were sorted based on the burst time. Whereas the shortest job first (SJF) policy does not favor task urgency. The tasks execution time would have been further

minimized if the communication latency was also been considered.

The model by [38] is also similar to that of Iqbal *et. al.* [37]. In their research, dynamic round-robin heuristic algorithm was used to significantly optimize tasks average turnaround, waiting and response times. The work is however deficient since burst time was also used as a determining yardstick. This does not cater for urgency; the priority was not based on deadlines.

Sultana *et. al.* [39] introduced a dynamic priority based scheduling algorithm to modify the existing static frameworks prior to their works using divisible load theory. Deadline was the deciding factors; tasks were arranged in their order of their deadlines whereby low deadlines tasks receive higher priority and the size of each of the tasks determines the most suitable resource. In this work, resources were classified master and slaves depending on their processing powers. Each task is checked against the master Virtual Machine (VM). If the VM speed can efficiently run a task, the task will be assigned to the Master. Otherwise, the task will be chunked such that other part is allocated to the slave. Their simulation results though showed that most tasks were executed before their deadlines but the fact that task were prioritized based on their deadlines does not guarantee a more minimized computational latency for the urgent tasks. Some tasks with low deadlines might be more forgiving (soft real-time) than high value deadlines tasks, the work failed to consider these fundamental distinctions.

Scheduling tasks in a resource rich environment like cloud also has improvement in User experience and Quality of Service QoS as parts of its goals. In order to achieve these goals, Wu *et. al.* [40] therefore proposed a precedence-based QoS tasks scheduling algorithm, which considered certain special attributes of tasks in the computation of their priority. These attributes include task privilege, task type, task length and task arrival time, these attributes are quantified and normalized to generate a priority weight or value for each of the tasks. After this, the tasks are thereafter arranged according to their priorities in batches. A matrix-based policy was employed to check for the resource which can give the earliest time to compute a particular task which was thereafter scheduled based on

their assigned priority. Although their CloudSim simulation showed a good performance and significant load balancing but the work is characterized with a long-distance latency due to scheduling at the cloud. Urgency of tasks was based on the context of commercial strength of the users and not necessarily based on criticality.

A resource and dependency-aware scheduling technique to offload multi-task for integrated edge-enabled IoV was put forward by Awada *et. al.* [41]. It introduced a unified Edge-Cloud structure, which ensures that resources at the edge run across Road-side Units and Base Stations. A single Control Plane aids the holistic monitoring of edge resources. The Control Panel helps to identify the availability of the infrastructure and based on this, tasks could be offloaded vis-à-vis their processing requirements. The scheduling technique involves intelligent policy of arranging multiple tasks and this eventually helped to achieve minimized execution time and better resource utilization. However, tasks dependency was only considered without task criticality.

Gabi *et. al.* [42] introduced mobile edge-cloud continuum using Fruit Fly-based Simulated Annealing Optimization Scheme (FSAOS). This framework serves as a potential solution for the associated issues when dynamic resource arrangements are required in a collaborative environment of edge and cloud. Meanwhile, in order to achieve balance between the local search and global search in the optimization scheme and to combat the premature convergence in the policy, simulated annealing was considered. The FSAOS provides proficient resource provisioning in Mobile Edge Clouds and at the same time meeting up with the user's and cloud providers' expectations. This model focuses on individual task rather than all tasks. This does not factor in the urgency of an individual tasks. Some non-latent friendly tasks receive lower execution time compared to other tasks.

An operating system strategy which was based on scheduling with a round robin policy was introduced by Noon *et. al.* [43]. The work presented an algorithm which utilized a new approach called dynamic-time-quantum; the idea of this approach is to make the operating systems adjusts the time quantum according to the burst time of the set of waiting processes in

the ready queue. The simulation result showed an improved round robin performance and the issue associated with the fixed time quantum was resolved. This model's flaw is that the elasticity of the time quantum was chosen based on burst time, and pre-emptive decision in the work was based on the execution time rather than task criticality.

A method based on the combination of Laxity and Ant Colony System (LBP-ACS) for cloud-fog task scheduling was put forward by Xu *et al.* [44]. The deadline and the priority of tasks were considered before scheduling of tasks was done in this policy. Also, laxity-based priority algorithm was adopted to cater for the delay sensitivity of tasks, and to prioritize scheduling sequence. The total energy consumption was minimized and the rate of tasks failure was also reduced when this policy was benchmarked against the existing scheduling policies. However, all tasks were offloaded to a remote environment whereas there could be some critical mobiles apps that could still be sufficiently and efficiently executed locally. This could have further beaten down the deadline and eliminate communication delay.

In their innovative study of mobile edge computing networks, Huang *et al.* [45] proposed a multi-server, multi-user and multi-task computation for offloading. They engaged a heterogeneous distributed deep learning-based offloading strategy which achieved a better convergence than the existing algorithms. Also, their policy showed a faster performance than the existing Relaxation-based strategies. Nevertheless, the model could pose more computational cost and application failure as MEC are limited in their resource compared to the Cloud Data center.

Mishra *et al.* [46] Introduced Energy-Efficient Deployment of Edge Datacenters for Mobile Clouds in Sustainable IoT, a task allocation framework that tries to save energy on both the edge device and the cloud side without forfeiting deadlines. Two execution environments were considered for computational offloading, the cloud or the edge. If the edge datacenter is computationally insufficient to run the task, the request gets kicked up to the cloud. They proposed a polynomial-time rule to juggle heterogeneous mobile applications across edge, cloud, or local execution.

Adding an edge datacenter as a middle layer seems to help. Energy consumption and execution time both dropped. The algorithm optimizes energy and delays reasonably well.

Nevertheless, the work does not offer much beyond meeting up with deadlines of task whereas satisfying the deadline does not translate to urgency of tasks. In addition, the work failed to give priority to the deadline-constrained tasks that are critical among the tasks under considerations.

Li *et al.* [47] examined energy-aware policy that involves sending Internet of Things-based tasks to heterogeneous networks. They framed energy minimization as a mixed-integer nonlinear program (MINLP), but an NP complete program. They resorted to split it into two smaller problems: Transmission Power Allocation (TPA) and the actual Computation Offloading Decision (COD). Then they iterated back and forth between the two. The paper factored in several real-world constraints: different energy consumption by the devices, permissible delay and resources. The results showed how resource distribution affects overall energy efficiency, and the approach optimizes both offloading and transmission power together to keep energy low. However, CPU cycles per second was the only computational resources that was considered despite the different types of heterogeneous computational resources i.e GPU. That is a very limited view of heterogeneity. Real edge nodes have GPUs, TPUs, and other accelerators. The authors acknowledged this and say future work will look at CPU and GPU mixes. But right now, the framework is too simplistic. The real problem is that computation is only allowed at edge devices. If an application is both delay-sensitive and computationally intensive, this framework has no good solution. It does not account for cases where the task is too heavy for the edge but too urgent to wait. That is a significant deficiency

Wang and Yu [48] devised a scheduling policy whereby less computationally demanding tasks are mapped to the resources with very high compute power, a min-min scheduling strategy. Tasks priority was based on the computational length. The policy would have been more sensitive had it been task scheduling yardstick was based on the tasks' urgency rather than the burst time. A short task does not necessarily imply an urgent task.

3.3 Deadline and Criticality

Future applications can easily leverage the resources at the edge layer when the cloud's computational servers are efficiently utilized and servicing some deadline-constrained tasks becomes faster. This was the focus of the model suggested by Lou *et. al.* [49]. The framework was researched to improve the makespan through task's successor's latest start time, servers' strength and bandwidth before scheduling was done. Still, the work failed to understand that some application deadlines are forgiving while some are stringent hence did not focus on prioritizing more critical tasks over others. Meeting up with the deadlines of the deadline constrained and computation intensive tasks actually mean reducing both the computational and transmission latency but does not factor in criticality of task. That a task is deadline-constrained and computation intensive does not represent its criticality. Some tasks are deadline sensitive but forgiving (soft-real time tasks). In this case, generically optimizing latency for all latency sensitive tasks without critically considering priorities can lead to undue delay for the latency sensitive critical mobile tasks among the deadline constrained tasks.

In a bid to solve task sequencing, resource searching, and mobility-aware scheduling problem, Lakhan *et. al.* [50] developed Deadline Aware and Energy-Efficient Scheduling (DEETS) algorithm for fine-grained tasks in mobile edge computing. This model considered two types of deadlines: hard deadlines and soft deadlines; whereas focus was only on the hard deadline of all applications. The scheduler could miss the soft deadline during scheduling in the MEC network. The primary motivation of the study targeted a reduced value of energy that will be consumed by MEC nodes and meets the deadline of applications. The shortcoming of this method are that it focused on hard deadline (critical) tasks but ignored soft deadlines and there is no fairness. Meanwhile, offloading was targeted at either the cloud or edge. No consideration was given to critical apps that could be ultra-low latent and at the same time less resource intensive which could easily be executed locally. In their decision making, criticality was the major factor, processing and latency indices were not considered.

Liu *et. al.* [51] posited a task offloading strategy for cloud-edge environments that keeps deadlines tight while minimizing energy and overall completion time. Their methodology engaged the firefly algorithm which uses two ideas: a position-based mapping that anchors each firefly to a stronger solution plus a linear movement scheme that replaces the standard update so fireflies can reposition more efficiently. The improvements are there, but the model still has significant gaps. Tasks are strictly atomic; no splitting across resources. For any task ζ_i , the model defines a tuple $\langle d_i, c_i \rangle$ and derives total workload w_i from it. Since every task runs independently, the framework has no way to exploit dependencies or give shorter jobs priority.

The scheduling is even more restrictive. Once a task starts on a server, that server is locked until the job finishes. This hard First Come First Served (FCFS) rule makes fairness difficult and leaves room for starvation, especially when a single long-running task camps on the server indefinitely.

Younesi *et. al.* [52] recently put forward MOSAIC- Mobility-Oriented Scheduling and Intelligent Resource Allocation for IoT . They identified that the existing scheduling frameworks have not been able to have a balance in optimizing some performance metrics such as low delay, energy proficiency in a real time scenario whereby there is a dynamics in device movement and heterogeneity in the workload. Attention was placed on the meeting up with deadline equally without reference to how forgiving a task might be. Although, all tasks could finish before their deadlines but if the Latency Sensitive Critical tasks (LSCTs) were given priority, the LSCTs would have been serviced earlier and consequently have a minimized or lesser response and turnaround times compared with their results.

Khan *et. al.* [53] formulated a Criticality-Aware Dynamic Task Scheduling Mechanism for Efficient Resource Load Balancing in Constrained Smart Manufacturing Environment. They identified that the existing frameworks paid slight attention to achieving reliable and robust service in smart manufacturing environment. With their strategy, production cost was reduced, smart factory resource utilization was enhanced leading to better

productivity. However, Fetching and execution of tasks were based on deadline values among the critical tasks. The work further used criticality bit (CB) as a yardstick to determine the criticality of a task; when a task carries CB = 1, it is declared critical, else, non-critical. As a snag, a task may have the minimum value of deadline but of low criticality degree. Therefore, a more sensitive framework which considers the actual nature of tasks (criticality) irrespective of its deadline would be more favorable to the latency sensitive critical tasks. Another weakness is that the task gave no technical or decisive explanation on how the criticality bit would be computed.

Rasouli *et. al.* [54] carried out a systematic literature review on resource management for mission-critical applications at the edge. They engaged a conservative methodology and looked at the service quality metrics that are important for urgent workloads in edge environments. They also flagged where prior reviews fall short. The paper is essentially a broad survey; it includes what's been done recently and what's still missing. They explained the performance metrics needed to keep user experience from falling apart when running critical applications at the edge. Yet, the work did not consider works that recognize the effect of transmission latency viz-a-viz the location where a particular task is run. Depending on location, WLAN, MAN or/and WAN latency are possible. The work did not consider offloading techniques and execution layer-based categorization reviews and there is no categorization by execution layer. That feels like a major gap because in edge computing, the whole game is deciding where to run a task, and that decision directly determines the latency profile. Ignoring offloading and execution-layer categorization means the review misses one of the most important levers for managing latency in mission-critical scenarios. Deadline-Aware Dynamic Task Scheduling in Edge-Cloud Collaborative Computing using a hierarchical architecture model for the Industrial Internet of Things (IIoT) was introduced by Zhang *et. al.* [55]. Mathematical formulation was put forward to minimize computational latency at the edge-cloud collaboration. The time it would take a task to be executed was predicted by considering the tasks' queue components such as the submission, waiting time and the resource compute power. It factors in each server's overall capability to pick a suitable one

for incoming tasks. The performance metric improvement are significant; Processing time drops 30%, cost drops 45%, and SLA violations fall 25% compared to recent alternatives. With performance evaluation comparison with the existing solutions, they achieved superior performances but urgency of tasks was not considered as a crucial factor.

“Performance-effective and low-complexity task scheduling for heterogeneous computing was researched by Topcuoglu *et. al.* [56]. They introduced Critical Path-on-Processor (CPOP) and Heterogeneous Earliest Finish Time (HEFT) algorithm. The two algorithms improved average running time and keep performance fairly stable. Critical jobs are mapped unto the resource to minimize the total execution time. They recorded improved performance and cost metrics in terms of average running time and stable performance. Truly the framework considered task priority with respect to processor selection but what qualifies a task to be critical was not pointed out by the authors.

Liu *et. al.* [57] proposed a Deadline-Constrained Multi-Objective Task Scheduling Algorithm in Mobile Cloud Environments. The algorithm outperformed the existing deadline-constrained strategies in terms of deadline success rate, service cost and mean load. Despite their better performance over the existing framework, the deadline-constraint finishing rate was done generically without giving cognizance to how to have more minimized response time for the critical tasks.

An algorithm to satisfy heterogeneous deadline-constrained tasks was put forward by Mahmud *et. al* [58]. Meanwhile, the delay sensitivity of tasks was also considered as an index of prioritization. Two algorithms were involved, one for the placement of application module and the second one was for the simplification of optimization problem in order to prevent resource idleness. The iFogSim simulation results of their policy suggested a superior performance when compared with other prior latency-based policies.

However, the proposed system lacked adequate provision which considered the mobility and customized settings of the users..

3.4 Delay sensitivity and service time minimization.

Ali and Iqbal [59] introduced a scheduling model of offloading microservice-based tasks in MCC to reduce computational cost and consumption of energy. It was proposed to run delay sensitive applications more than the previous works using Task Scheduling and Microservice-based Computational Offloading (TSMCO) framework. Tactlessly, First-Come-First-Served scheduling policy was however engaged. The implication of this strategy is that even if an application is a latency sensitive job but arrive late and has a long execution time, it suffers and meeting up with the user specified deadline would be defeated.

Pang *et. al* [60] Leveraged Deep Reinforcement Learning (DRL) to minimize average tasks processing time in satellite mobile edge computing. The heterogeneity of both jobs and that of the sat-Mobile Edge Cloud servers were considered to achieve a better optimizations metrics. When compared with random algorithm, greedy algorithm and genetic algorithm, the reinforcement algorithm achieved a reduced percent of 22.1, 30.6, and 41.3%, respectively. Despite all these improved performance metrics, minimizing the average service time of tasks generically creates undue computational latency for the critical tasks.

Prior to their work, Han *et. al* [61] identified that most existing frameworks considered tasks dispatching based on the foreknowledge of the arrival times of offloaded tasks. Thus, they pointed out that the stochastic consideration could be unfit for the task arrival distribution in a real-time environment like cloud. Their strategy is a probabilistic framework which ignores the fore-knowledge of the tasks arrival before deciding where a particular task is to be run, either locally or at the edge using latency sensitivity as the deciding factors. The algorithm however failed to recognize that some tasks are latent unfriendly but non-critical while some are both critical and at the same time, latency sensitive. Issue with this technique could mean that on individual basis, critical tasks that are relatively of high latency values could suffer additional and avoidable long latency.

An online scheduling method was proposed by Chen *et. al.* [62] to minimize the penalty incurred by task tardiness in real-time systems.

The authors assumed MEC environment as a soft real-time system. Their approach addresses both task-to-server assignment and task scheduling on the server. They subscribed to a heuristic algorithm that prioritizes tasks based on the urgency. Its aim was to reduce losses caused by task timeouts. Results from experiments carried out showed that their algorithm outperforms classic real-time task scheduling algorithms. However, this paper did not consider task dependencies; instead, all tasks were treated as independent and yardsticks to determine the urgency of tasks were not clearly spelt out. This is not sensitive enough.

Almutairi *et. al.* [63] leveraged Integer Linear Programming strategy to reduce the overall network and processing times of Unmanned Aerial Vehicles in edge-cloud computing domain. When the results of their simulation was compared with other strategies, their offloading strategy reduced the service time at the cloud and edge by 55% and 35% respectively. However, cognizance was not given to the nature (criticality) of the application coming from each of the unmanned Aerial Vehicles (Drone). In addition, the policy's objective could be defeated since all applications are offloaded without considering that some apps with low battery requirement, low latency and computational requirement can still be executed locally. This amounts to poor resource utilizations.

Wang *et. al* [64] proposed a cooperative multi-tasks scheduling based on the Ant Colony Optimization algorithm (CMSACO) to achieve optimum profit with constraints of finish time and resource capacity. Yet, the work did not consider attending to users based on the urgency of their tasks.

Sahito and Kehar [65] factored in the impacts of some contents of mobile networks such as bandwidth and network status that changed dynamically on computational offloading of tasks. When evaluated against the baseline models, their strategy which involved a dynamic adjustment policy outperformed the existing tasks offloading models in terms of minimization of total service time and optimal availability of service. Additional components were added to the current Mobile Cloud Computing (MCC) architecture to allow user mobility. Response time was minimized with this policy. Of course, the overall service time

was claimed to be significantly reduced but it is generically improved without any preference to how critical a task is. Meanwhile, another major drawback of the work is the non-inclusion and lack of consideration of edge environment which would have been to further minimize the transmission latency of the delay sensitive as it brings better compute power closer to the edge of a networks

A novel scheduling algorithm using a Directed Acyclic Graph (DAG) based on the prediction of the tasks computation time algorithm (PTCT) was put forward by Al-Maytami *et. al.* [66]. This model integrates Principle Components Analysis (PCA) in order to reduce the Expected Time to Compute (ETC) matrix. The scheduling policy did a thorough examination of the incoming tasks before efficient resource allocation is done. This therefore gave rise to a superior performance achievement in terms of reduced makespan and simplicity. PTCT method decreases both of these costs noticeably but failed to prioritize tasks with urgency such that any failure to meet with their deadline could cause loss of lives and properties.

Kuppusamy *et. al.* [67] introduced a reinforced strategy Dynamic Opposition Learning Based Social Spider Optimization (DOLSSO) algorithm to enhance individual superiority and schedule workflow in fog computing. SSO algorithm mimics the foraging actions of vibration sense. Prey or another social spider formed the food source. The vibration sense determined the direction to the source of the food, the search space being the web while the feasible solution was the spider. When compared with five existing metaheuristics, their simulation results showed a minimized energy consumption of about 5-10% and 10% - 15% better CPU utilization. The work did not however consider additional performance

metrics such as deadlines and criticality of factors were not given attention.

An Internet of Things scheduling models to minimize latency in hybrid fog-cloud computing was proposed by Aburukba *et. al.* [68]. Genetic algorithm was customized and implemented heuristically to map the requests generated by Internet of Things devices onto fog and cloud environments. The customized technique was geared towards latency minimization and servicing the tasks earlier before their deadlines. When compared with the traditional scheduling algorithms such as Round Robin and weighted Fair Queuing, the strategy performed better in terms of the delay minimization and deadlines success rate by 21.9% and 31% respectively. The drawback of the heuristics is that it lacks critical scheduling, and benchmarking the results of modern scheduling frameworks with traditional scheduling algorithms is not standard and sufficient. Poor resource utilization also characterized this study.

A decision scheduling for cloud computing tasks relying on solving large linear systems of equations was put forward by He [69]. Scheduling of parallel tasks in this policy leveraged data mining and Particle Swarm Optimization strategies. They contributed to knowledge by ensuring that the expectations of users were met in terms of completion time, cost and bandwidth, but also optimize the cost. Latency and cost optimization would have been more robust if the policy gave cognizance to the urgency of tasks among the tasks under investigation.

4. The Thematic Organization Analysis and Results.

The ongoing research involved the review of a lot of papers, 42 of which are used for this review paper and the analysis are broken down as follows:

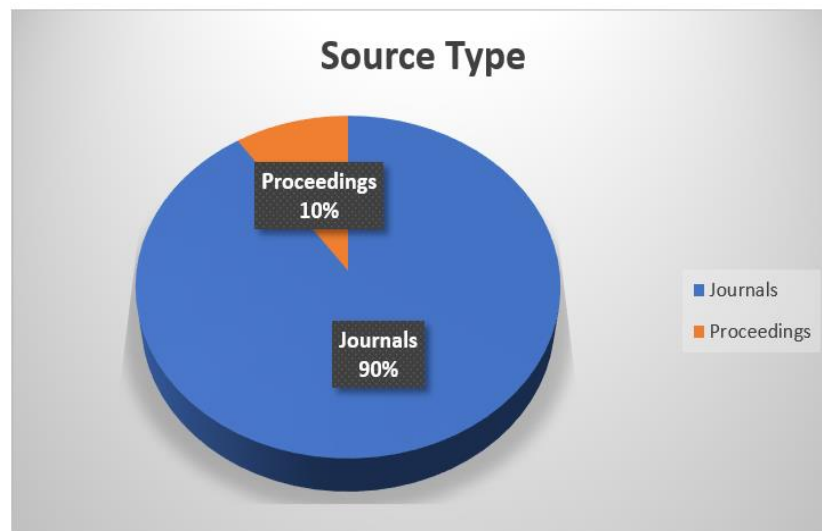


Figure 1 :The reviewed papers analysis by academic resource type

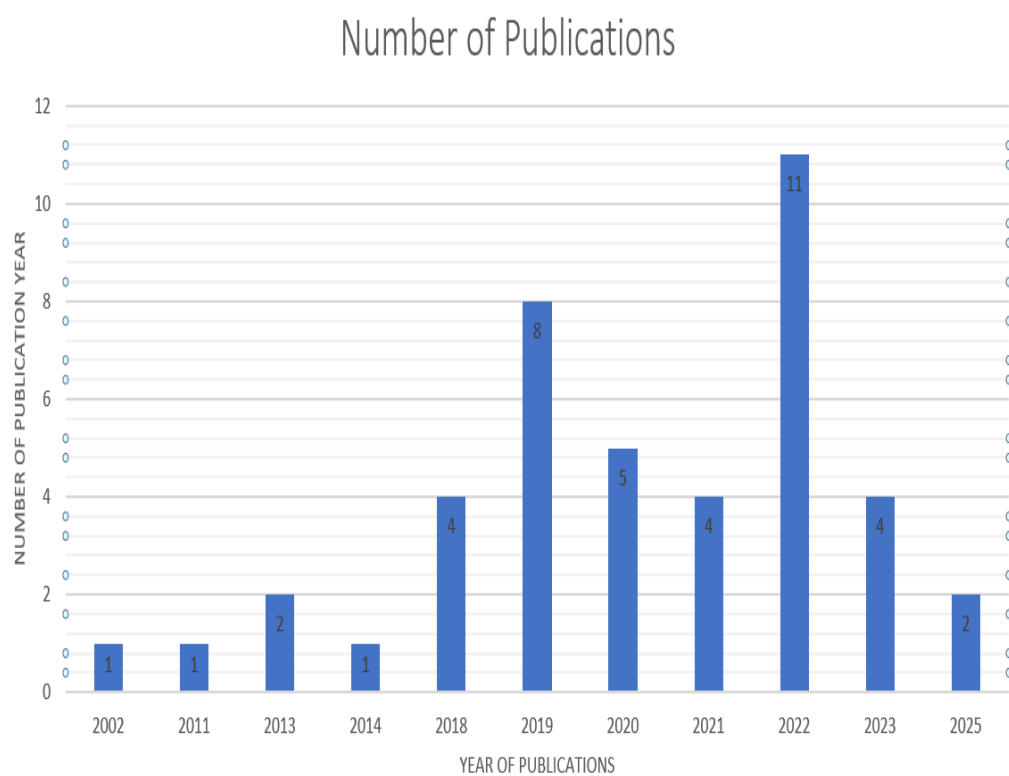


Figure 2 : The reviewed papers analysis by year of publication

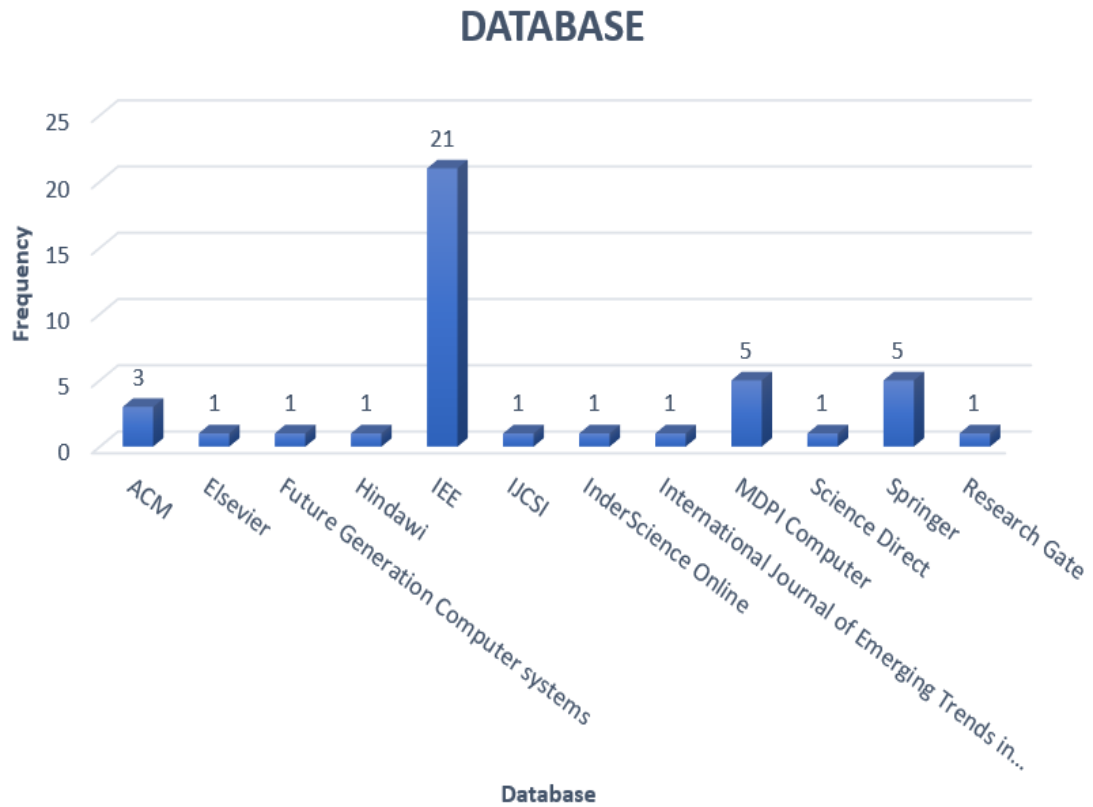


Figure 3 : The Reviewed Paper Analysis by database

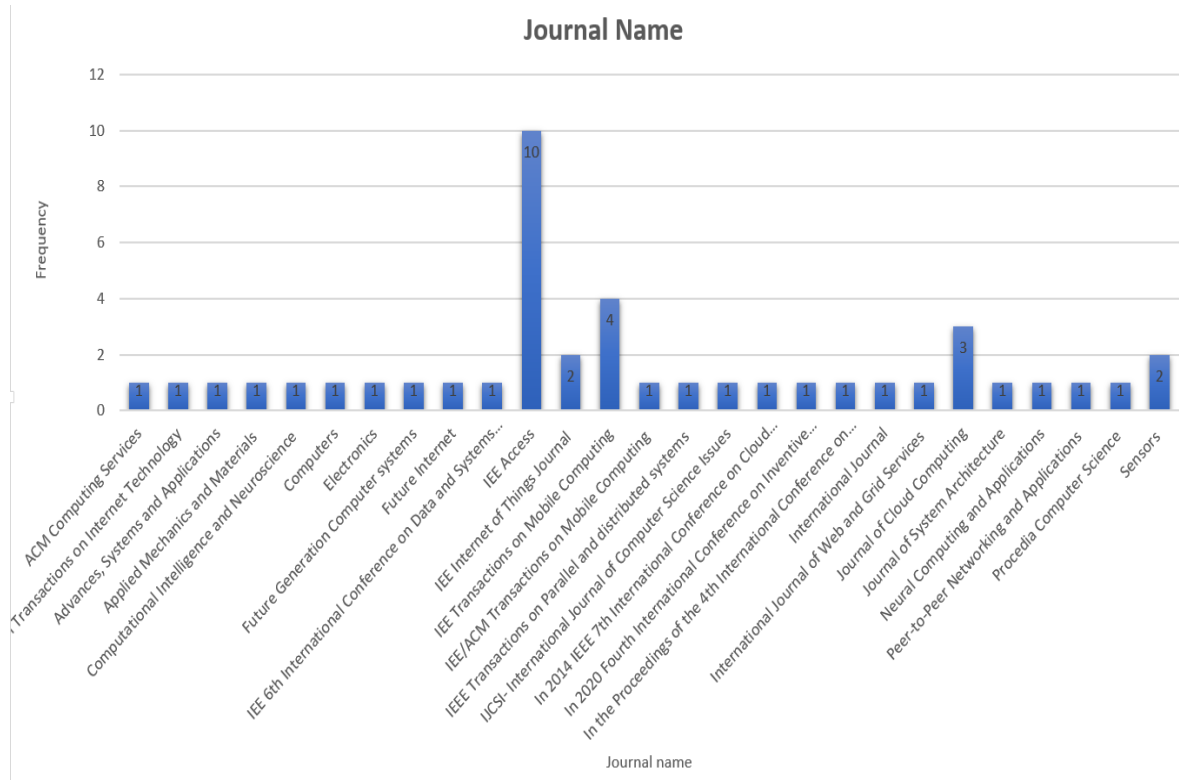


Figure 4: The reviewed Papers by Journal Name

Discussion and interpretation

In Figure 1, journal papers unarguably have the lion share among the papers reviewed with about 90%. This points to the fact that the cited researchers in this paper must have gone through rigorous peer reviews before the validation of their works. The graph corroborates the credibility, the expertise and professional reliability in this review paper. In addition, it points to the extensiveness and reproducibility of the works cited in this paper.

In Figure 2, about 18 out of 42 reviewed works are from 2022 to 2026, this is a significant portion and as a benefit, it establishes the academic freshness and contemporariness in the review paper. The up-to-date content could help readers to be familiar with the recent trends and save them from the dangers of reinventing what have been recently published

Analysis of the Review Papers by Database

To be premier is quality, it is professionalism and it is standard. Institute of Electrical and Electronics Engineers Xplore (IEEE Xplore) is a premier electronic database which allows access to academic works that have undergone thorough and reproducible stages. It is a leading and reputable virtual database serving as a repository and academic library to millions of world researchers. From the Figure 3, IEEE electronic database account for 50% of reviewed works in this paper. This suggests the reliability and freedom from academic biasness of this academic review. Meanwhile, as it is obvious in figure 4 that most of the journals and proceedings are archived in IEEE Xplore, this does not only point to confidence in this paper but also promotes quality academic outcomes and soundness when leveraged by researchers.

5. Limitation and Future Scope of Research

About 42 papers were reviewed in this article which spanned from 2002 to 2026. However only 5 papers were reviewed between 2002 and 2014. Although, it is an improvement on the existing review paper but a lot can still be done. Future work can be extended to having more paper within these years as this would be richer and more informative to the research community. Papers that consider scheduling and prioritizing critical tasks in edge-cloud environment dynamically in the presence of real-time combinatorial constraints can be more meaningful in the future work. Nevertheless, this

paper has unveiled a lot of research gaps existing in the current technologies related to delay sensitive IoT and UAV mobile critical tasks and lucrative research insight to future advances.

6. Conclusion

In this paper, the review of papers that are connected with delay sensitivity and computationally intensive mobile tasks has been presented. The gap identified by the ongoing research study is presented in the abstract and the introductory session introduces readers to some key concepts in the topic under review such as smart devices, edge and cloud computing paradigms alongside their strengths and limitations. The review methodology was sectioned according to four different themes among the reviewed papers such as the execution layer, resource allocation determiner, the delay sensitivity and service time minimization and deadline and criticality.

Summary, to present at a glance the strength, the contribution and the identified gap in each of the reviewed papers are presented. This paper would not only give research community a soft landing in contribution to knowledge but at the same time give a clear and current overview on the trend and state of research to industrial practitioners on scheduling involving IoT and UAV based critical and latency sensitive tasks.

References

- [1] Islama, A., Debnath, A., Ghose, M., & Chakraborty, S. (2021). A survey on task offloading in multi-access edge computing. *Journal of Systems Architecture*, 118, 102225. <https://doi.org/10.1016/j.sysarc.2021.102225>.
- [2] Zissis, D. and Lekkas, D. (2012). Addressing cloud computing security issues. *Future Generation Computer Systems*, 28(3), pp.583–592. doi:10.1016/j.future.2010.12.006.
- [3] Jaber, A., Mohammad, A., Hatem A. A., Barzan A. Y. , & Jaafar, M. H. E. (2022) .Delay-Optimal Task Offloading for -Enabled Edge-Cloud Computing Systems. *IEEE Open Access Journal*, 2022.
- [4] Peter M., Timothy G., (2011) “The NIST Definition of Cloud Computing” [http://docs.ismgcorp.com/files/external/Draft-SP-800-145 Cloud Definition.pdf](http://docs.ismgcorp.com/files/external/Draft-SP-800-145%20Cloud%20Definition.pdf).
- [5] Ali, M. (2009). *Green cloud on the horizon*. In: *Proceedings of the IEEE International Conference on Cloud Computing*. Berlin and Heidelberg: Springer Berlin Heidelberg,

- pp.451–459. doi:10.1007/978-3-642-10665-1_41.
- [6] Shahryari, O.K., Pedram, H., Khajehvand, V. and TakhtFooladi, M.D. (2020). Energy-efficient and delay-guaranteed computation offloading for fog-based IoT networks. *Computer Networks*, 182, p.107511. doi:10.1016/j.comnet.2020.107511.
- [7] Ferrer, A.J., Marquès, J.M. and Jorba, J. (2019). Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing. *ACM Computing Surveys*, 51(6), pp.1–36. doi:10.1145/3243929.
- [8] Aburukba, R.O., AliKarrar, M., Landolsi, T. and El-Fakih, K. (2020). Scheduling Internet of Things requests to minimize latency in hybrid Fog–Cloud computing. *Future Generation Computer Systems*, 111, pp.539–551. doi:10.1016/j.future.2019.09.039.
- [9] Sun, Y., Zhou, S. and Xu, J. (2017). EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks. *IEEE Journal on Selected Areas in Communications*, 35(11), pp.2637–2646. doi:10.1109/JSAC.2017.2760160.
- [10] Redowan Mahmud, Kotagiri Ramamohanarao, and Rajkumar Buyya. 2018. Latency-Aware Application Module Management for Fog Computing Environments. *ACM Trans. Internet Technol.* 19, 1, Article 9 (November 2018), 21 pages. <https://doi.org/10.1145/318659>.
- [11] Dastjerdi, A.V., Gupta, H., Calheiros, R.N., Ghosh, S.K. and Buyya, R. (2016). Fog computing: Principles, architectures, and applications. In: *Internet of Things*. Morgan Kaufmann, pp.61–75. doi:10.1016/B978-0-12-805395-9.00004-6.
- [12] OpenFog, C. (2017). Openfog reference architecture for fog computing. URL: <https://www.openfogconsortium.org/ra>.
- [13] Mahmud, R., Ramamohanarao, K., and Buyya, R. (2018). Latency-aware application module management for fog computing environments. *ACM Transactions on Internet Technology (TOIT)*, 19(1), 1-21.
- [14] De, D., Mukherjee, A. and Guha Roy, D. (2020). Power and delay efficient multilevel offloading strategies for mobile cloud computing. *Wireless Personal Communications*, 112, pp.2159–2186. doi:10.1007/s11277-020-07144-1.
- [15] Belal Ali Al-maytami 1,2, Pingzhi Fan 1, Abir Hussain 3, Thar Baker 3, And Panos Liatsis, (2019). Institute Of Mobile Communication, Southwest Jiaotong University, Chengdu 611756, China. A Task Scheduling Algorithm With Improved Makespan Based On Prediction Of Tasks Computation Time Algorithm For Cloud Computing.
- [16] Demchenko, Y. and De Laat, C. (2011). Defining generic architecture for cloud infrastructure as a service model. In: *The International Symposium on Grids and Clouds and the Open Grid Forum*. Academia Sinica, pp.2–10. doi:10.22323/1.133.0115.
- [17] Mishra, S. K., Puthal, D., Sahoo, B., Sharma, S., Xue, Z., and Zomaya, A. Y. (2018). Energy-efficient deployment of edge datacenters for mobile clouds in sustainable IoT. *Ieee Access*, 6, 56587-56597.
- [18] Chaogang, T., Xianglin, W., Shuo, X., Wei, C., Weidong, F., Wuxiong, Z., & Mingyang, H. (2018). A mobile cloud-based scheduling strategy for industrial internet of things. School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221000, China 2Nanjing Telecommunication Technology Research Institute, Nanjing 210000, China 3Key Laboratory of Wireless Sensor Network and Communication, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China Corresponding author: Wei Chen (chenw@cumt.edu.cn).
- [19] Guo, S., Xiao, B., Yang, Y. and Yang, Y. (2016). Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In: *Proceedings of the IEEE INFOCOM 2016 – The 35th Annual IEEE International Conference on Computer Communications*. IEEE, pp.1–9. doi:10.1109/INFOCOM.2016.7524497.
- [20] Wang, X., Wang, K., Wu, S., Di, S., Yang, K. and Jin, H. (2016). Dynamic resource scheduling in cloud radio access network with mobile cloud computing. In: *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*. IEEE, pp.1–6. doi:10.1109/IWQoS.2016.7590408.
- [21] Slim, B. (2020). Scheduling of Dependent Tasks with Probabilistic Execution Times on Multi-core Processors. *Hardware Architecture [cs.AR]*. Sorbonne Université. English. ffNNT : 2020SORUS188ff. fftel-03404155v3f.
- [22] Kopetz, H. and Steiner, W. (2022). *Real-Time Systems: Design Principles for Distributed Embedded Applications*. 3rd ed. Cham: Springer. doi:10.1007/978-3-031-11992-7.
- [23] Li et al. *J Wireless Com Network* -- <https://doi.org/10.1186/s13638-021-01941-3>.
- [24] Braik, M.S. (2021). Chameleon Swarm Algorithm: A bio-inspired optimizer for solving engineering design problems. *Expert Systems with Applications*, 174, p.114685. doi:10.1016/j.eswa.2021.114685.

- [25] Rasouli, N., Klein, C., and Elmroth, E. (2025). Resource management for mission-critical applications in edge computing: systematic review on recent research and open issues. *ACM Computing Surveys*, 58(3), 1-37.
- [26] Samanta, A., and Chang, Z. (2019). Adaptive service offloading for revenue maximization in mobile edge computing with delay-constraint. *IEEE Internet of Things Journal*, 6(2), 3864-3872.
- [27] Atan, B., Basaran, M., Calik, N., Basaran, S. T., Akkuzu, G., and Durak-Ata, L. (2022). AI-empowered fast task execution decision for delay-sensitive IoT applications in edge computing networks. *IEEE Access*, 11, 1324-1334.
- [28] Naouri, A., Wu, H., Nouri, N. A., Dhelim, S., and Ning, H. (2021). A novel framework for mobile-edge computing by optimizing task offloading. *IEEE Internet of Things Journal*, 8(16), 13065-13076.
- [29] Zhang, X. (2023). A fine-grained task scheduling mechanism for digital economy services based on intelligent edge and cloud computing. *Journal of Cloud Computing: Advances, Systems and Applications*, 12(1), Article 30. <https://doi.org/10.1186/s13677-023-00402-0>
- [30] Li, L., and Zhang, H. (2020). Delay optimization strategy for service cache and task offloading in three-tier architecture mobile edge computing system. *IEEE Access*, 8, 59906-59921. <https://doi.org/10.1109/ACCESS.2020.2982992>
- [31] Lin, X., Wang, Y., Xie, Q., and Pedram, M. (2014, June). Energy and performance-aware task scheduling in a mobile cloud computing environment. In *2014 IEEE 7th international conference on cloud computing* (pp. 192-199). IEEE.
- [32] Goudarzi, M., Wu, H., Palaniswami, M., and Buyya, R. (2020). An application placement technique for concurrent IoT applications in edge and fog computing environments. *IEEE Transactions on Mobile Computing*, 20(4), 1298-1311.
- [33] Yi, C., Cai, J., and Su, Z. (2019). A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications. *IEEE Transactions on Mobile Computing*, 19(1), 29-43.
- [34] Oguntunde Toyin. (2012). The Quality of Transmission Media, a Determinant of network Throughput : Nigeria Unshielded Pair (UTP) Cable-Market Study. *Computing, Information System & Development informatics Journal*. Vol 3, No.3. pp 77-82. Online at www.cisdijournal.net
- [35] Almutairi, J., and Aldossary, M. (2021). A novel approach for IoT tasks offloading in edge-cloud environments. *Journal of cloud computing*, Springer 10(1), 28.
- [36] Lakhan, A., Mohammed, M. A., Abdulkareem, K. H., Jaber, M. M., Nedoma, J., Martinek, R., and Zmij, P. (2022a). Delay optimal schemes for Internet of Things applications in heterogeneous edge cloud computing networks. *Sensors*, 22(16), 5937.
- [37] Iqbal, M., Ullah, Z., Khan, I. A., Aslam, S., Shaheer, H., Humayon, M., ... and Mehmood, A. (2023). Optimizing task execution: the impact of dynamic time quantum and priorities on round robin scheduling. *Future Internet*, 15(3), 104.
- [38] Alhaidari, F., & Balharith, T. Z. (2021). Enhanced round-robin algorithm in the cloud computing environment for optimal task scheduling. *Computers*, 10(5), 63.
- [39] Sultana, S. R., Meena, K. S., Alaguvathana, P., and Abinaya, K. (2020, January). Priority based scheduling algorithm using divisible load theory in cloud. In *2020 Fourth International Conference on Inventive Systems and Control (ICISC)* (pp. 771-776). IEEE.
- [40] Wu, X., Deng, M., Zhang, R., Zeng, B., and Zhou, S. (2013). A task scheduling algorithm based on QoS-driven in cloud computing. *Procedia Computer Science*, 17, 1162-1169.
- [41] Awada, U., Zhang, J., Chen, S., Li, S., and Yang, S. (2023). Resource-aware multi-task offloading and dependency-aware scheduling for integrated edge-enabled IoV. *Journal of Systems Architecture*, 141, Article 102923. <https://doi.org/10.1016/j.sysarc.2023.102923>
- [42] Gabi, D., Dankolo, N. M., Muslim, A. A., Abraham, A., Joda, M. U., Zainal, A., and Zakaria, Z. (2022). Dynamic scheduling of heterogeneous resources across mobile edge-cloud continuum using fruit fly-based simulated annealing optimization scheme. *Neural Computing and Applications*, 34, 14085-14105. <https://doi.org/10.1007/s00521-022-07260-y>.
- [43] Noon, A., Kalakech, A., and Kadry, S. (2011). A new round robin based scheduling algorithm for operating systems: Dynamic quantum using the mean average. *IJCSI International Journal of Computer Science Issues*, 8(3, No. 1), 224-229. <https://www.ijcsi.org/papers/IJCSI-8-3-1-224-229.pdf> (or <https://arxiv.org/abs/1111.5348>).
- [44] Xu, J., Hao, Z., Zhang, R., and Sun, X. (2019). A method based on the combination of laxity and ant colony system for cloud-fog task scheduling. *IEEE access*, 7, 116218-116226.
- [45] Huang, L., Feng, X., Zhang, L., Qian, L., and Wu, Y. (2019). Multi-Server Multi-User Multi-

- Task Computation Offloading for Mobile Edge Computing Networks. *Sensors*, 19(6), 1446. <https://doi.org/10.3390/s19061446>
- [46] Mishra, S.K., Puthal, D., Rodrigues, J.J., Sahoo, B. and Dutkiewicz, E. (2018). Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications. *IEEE Transactions on Industrial Informatics*, 14(10), pp.4497–4506. doi:10.1109/TII.2018.2791619.
- [47] Li, S., Tao, Y., Qin, X., Liu, L., Zhang, Z., and Zhang, P. (2019). Energy-aware mobile edge computation offloading for IoT over heterogenous networks. *IEEE Access*, 7, 13092–13105.
- [48] Wang, G., and Yu, H. C. (2013). Task scheduling algorithm based on improved Min-Min algorithm in cloud computing environment. *Applied Mechanics and Materials*, 303, 2429–2432.
- [49] Lou, J., Tang, Z., Zhang, S., Jia, W., Zhao, W., and Li, J. (2022). Cost-effective scheduling for dependent tasks with tight deadline constraints in mobile edge computing. *IEEE Transactions on Mobile Computing*, 22(10), 5829–5845.
- [50] Lakhan, A., Mohammed, M. A., Rashid, A. N., Kadry, S., and Abdulkareem, K. H. (2022b). Deadline-aware and energy-efficient scheduling algorithm for fine-grained tasks in mobile edge computing. *International Journal of Web and Grid Services*, 18(2), 168–193.
- [51] Liu, Y., Wang, X., Cheng, M., Wang, J., and Zhang, Y. (2020, December). An efficient task offloading strategy in cloud-edge computing under deadline constraints. In 2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS) (pp. 661–667). IEEE.
- [52] Younesi, A., Toghiani, M., Safari, S., Ansari, M., and Fahringer, T. (2025). MOSAIC: Mobility-Oriented Scheduling and Intelligent Resource Allocation for IoT. *IEEE Transactions on Mobile Computing*.
- [53] Khan, A. N., Iqbal, N., Rizwan, A., Malik, S., Ahmad, R., and Kim, D. H. (2022). A criticality-aware dynamic task scheduling mechanism for efficient resource load balancing in constrained smart manufacturing environment. *IEEE Access*, 10, 50933–50946.
- [54] Rasouli, N., Klein, C., and Elmroth, E. (2025). Resource management for mission-critical applications in edge computing: systematic review on recent research and open issues. *ACM Computing Surveys*, 58(3), 1–37.
- [55] Zhang, Y., Tang, B., Luo, J., and Zhang, J. (2022). Deadline-aware dynamic task scheduling in edge-cloud collaborative computing. *Electronics*, 11(15), 2464.
- [56] Topcuoglu, H., Hariri, S., and Wu, M. Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE transactions on parallel and distributed systems*, 13(3), 260–274.
- [57] Liu, L., Fan, Q., and Buyya, R. (2018). A deadline-constrained multi-objective task scheduling algorithm in mobile cloud environments. *IEEE Access*, 6, 52982–52996.
- [58] Mahmud, R., Ramamohanarao, K. and Buyya, R. (2018). Latency-aware application module management for fog computing environments. *ACM Transactions on Internet Technology*, 19(1), pp.1–21. doi:10.1145/3186592.
- [59] Ali, A., and Iqbal, M. M. (2022). A cost and energy efficient task scheduling technique to offload microservices based applications in mobile cloud computing. *IEEE Access*, 10, 46633–46651.
- [60] Pang, S., Zheng, J., Wang, M., Qiao, S., He, X., and Gao, C. (2023). Minimize average tasks processing time in satellite mobile edge computing systems via a deep reinforcement learning method. *Journal of Cloud Computing*, 12(1), 159.
- [61] Han, Z., Tan, H., Li, X. Y., Jiang, S. H. C., Li, Y., and Lau, F. C. (2019). OnDisc: Online latency-sensitive job dispatching and scheduling in heterogeneous edge-clouds. *IEEE/ACM Transactions on Networking*, 27(6), 2472–2485.
- [62] Chen, X., Xu, H., and Huang, L. (2019, May). Online scheduling strategy to minimize penalty of tardiness for real-time tasks in mobile edge computing systems. In *Proceedings of the 4th International Conference on Big Data and Computing* (pp. 107–114).
- [63] Almutairi, J., Aldossary, M., Alharbi, H. A., Yosuf, B. A., and Elmighani, J. M. (2022). Delay-optimal task offloading for UAV-enabled edge-cloud computing systems. *IEEE Access*, 10, 51575–51586.
- [64] Wang, L. Jiao, J. Li, and M. Muhlh . (2017). Online Resource Allocation “ for Arbitrary User Mobility in Distributed Edge Clouds,” in *Proc. of IEEE ICDCS*.
- [65] Sahito, M. A., and Kehar, A. (2021). Dynamic content enabled microservice for business applications in distributed cloudlet cloud network. *International Journal*, 9(7), 1035–1039.
- [66] Al-Maytami, B. A., Fan, P., Hussain, A., Baker, T., and Liatsis, P. (2019). A task scheduling algorithm with improved makespan based on prediction of tasks computation time algorithm

- for cloud computing. *IEEE Access*, 7, 160916-160926.
- [67] Kuppusamy, P., Kumari, N. M. J., Alghamdi, W.Y., Alyami, H., Ramalingam, R., Javed, A. R., and Rashid, M. (2022). Job scheduling problem in fog-cloud-based environment using reinforced social spider optimization. *Journal of Cloud Computing*, 11(1), 99.
- [68] Aburukba, R.O., AliKarrar, M., Landolsi, T. and El-Fakih, K. (2020). Scheduling Internet of Things requests to minimize latency in hybrid Fog-Cloud computing. *Future Generation Computer Systems*, 111, pp.539–551. doi:10.1016/j.future.2019.09.039.
- [69] He, J. (2022). Decision scheduling for cloud computing tasks relying on solving large linear systems of equations. *Computational Intelligence and Neuroscience*, 2022(1), 3411959.