A Weighted Scoring Model of Heuristic-Based Workload Scheduling Approaches in Edge-Cloud Environments

Hasnae NOUHAS[1], Abdessamad BELANGOUR[2], Mahmoud NASSAR[3]
Laboratory of Artificial Intelligence and Systems LAIS-Faculty of Sciences Ben M'Sik, Hassan II University, Casablanca, Morocco[1, 2]
IMS Team-ADMIR Laboratory-ENSIAS-Rabat IT Center, Mohammed V University in Rabat, Rabat, Morocco[3]

***Abstract*—Hybrid edge–cloud computing has emerged as a promising paradigm to meet the demands of latency-sensitive and resource-aware applications by combining the low-latency benefits of edge nodes with the scalability of cloud infrastructure. Efficient workload scheduling in such environments remains a critical challenge due to the heterogeneity of resources, dynamic network conditions, and diverse application requirements. This paper presents a comprehensive survey and comparative analysis of heuristic and metaheuristic scheduling algorithms tailored for edge–cloud systems. Seven representative algorithms, including Greedy Resource-Aware Heuristics (GRAH), Heterogeneous Earliest Finish Time (HEFT), Min-Min/Max-Min, Genetic Algorithm, Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Tabu Search, are evaluated against seven key criteria: latency awareness, energy efficiency, scalability, scheduling accuracy, implementation complexity, resource utilization, and adaptability. The evaluation is literature-driven and structured through a Weighted Scoring Model (WSM), which synthesizes findings from prior simulation-based and experiment-based studies into a comparative framework. Results indicate that Greedy Resource-Aware Heuristics offer the best trade-off for real-time, dynamic scenarios, while optimization-based methods, like GA and Tabu Search, provide superior accuracy and resource balance at the cost of increased complexity. The findings highlight critical trade-offs and offer guidance on selecting appropriate scheduling strategies based on application-specific goals and system constraints.**

Keywords*—Edge computing; cloud computing; workload scheduling; heuristic algorithms; metaheuristics; scheduling optimization; ACO; PSO; HEFT; Tabu Search; GA; Greedy Resource-Aware Heuristics; Min-Min; Max-Min; WSM; Weighted Scoring Model

I. Introduction

With each new breakthrough digital tech innovation, they're driving one revolution after another of intelligent services that become increasingly integrated into our lifestyle and industrial workflow processes. Consider just real-time video surveillance, intelligent transport systems, self-driving cars, robot-assisted manufacturing, and telemedicine; each of these very different use cases have one thing in common; they need fast, predictable, contextual computing. They tend to operate in very dynamic environments, process very large data inputs, and make decisions within fractions of a second.

Cloud computing has always made data processing possible by virtue of large-scale computing and large-scale storage. But it is not optimized with ultra-low latency for use cases today that are latency-aware. Because of clouds' data centers' locations being far from end-users, communicating across those long-haul distances, paired with bandwidth constraints, becomes a serious chokepoint [1], [2]. Just running on the cloud is not good enough for use cases when milliseconds do matter at the end.

In an endeavor to close that gap, we are headed towards hybrid edge–cloud architectures. Here, processing of work is judiciously split; lighter workloads are processed close to or at the edge of the network (i.e. close to the place of data origination), while high-intensity workloads are shifted out to the cloud [2], [3]. Both of them together offer: edge computing's speed and proximity advantage as well as scalability and power of the cloud. That's the perfect destination for new services where a very fast response time as well as high compute capacity are an imperative.

But to actually make this architecture work well together, however, one of its needs for capabilities is intelligent scheduling of work. That is, decisions at the time of where and when each of its pieces of work needs to run; out at the edge, out at the cloud, or across two. Decisions of how they are made have huge effects on system performance, from response time through energy efficiency, all the way to resource utilization and utilization of networks [3]. That is, scheduling of workloads is not after-the-fact business at all; it's utterly essential for successful hybrid computing.

But coming up with an effective schedule solution for such very dynamic and very complex edge–cloud environments is no small feat. Schedulers are faced with many things happening simultaneously, like limited edge resources, fluctuating network conditions, mobility devices, tasks with deadline constraints, and user preferences. And sometimes they need to do all of that on the fly. Traditional optimization approaches, although accurate, are too resource-intensive and latency-prone for real-time programs, especially large-scale programs [3]. That's why there's greater emphasis on heuristic-based schedule approaches; a faster, more agile approach that's better adapted to today's edge-cloud computing.

Heuristics provide fast, light-weight, and flexible solutions that best suit edge–cloud applications. They do not guarantee an optimal solution but provide a fast, good-enough solution that is precisely desired in real-world applications. Min-Min, Max-Min, Heterogeneous Earliest Finish Time (HEFT), and Greedy Resource-Aware are intuitive decision-making or rule-of-thumb–based schemes that are simple to implement as well as fine-tune. Although seemingly extremely trivial, such heuristics could possibly handle multi-dimensional trade-offs like latency vs. energy consumption or edge load balancing vs. cloud optimality with remarkable efficiency [4].

However, the sheer variety of heuristic methods—and the lack of standardized benchmarks—makes it difficult for researchers and practitioners to choose the most appropriate algorithm for a given application. Some heuristics are tailored for low latency, while others excel at resource efficiency or scalability. As a result, there is a growing need for a unified, multi-criteria evaluation framework that enables a fair and meaningful comparison across these approaches. This motivates our work: to provide a comprehensive review and comparative analysis of heuristic-based scheduling methods, supported by a Weighted Scoring Model (WSM) that captures the practical trade-offs involved in real-world deployments.

This paper aims to bridge the gap between the growing number of heuristic-based scheduling algorithms and the lack of a standardized, comparative evaluation framework tailored for hybrid edge–cloud environments. Unlike works that focus on individual algorithms, the novelty of our study lies in providing a unified, multi-criteria evaluation framework through the Weighted Scoring Model (WSM). We do not propose a fundamentally new scheduling algorithm in this paper; rather, we offer a systematic and transparent synthesis that highlights trade-offs, strengths, and weaknesses across existing approaches. In this way, the paper provides both practitioners and researchers with a practical decision-support tool and a baseline for identifying research gaps. Future work will build on this foundation to design and validate novel adaptive scheduling methods tailored to real-world, large-scale edge–cloud deployments. Our contributions can be summarized as follows:

- Comprehensive Survey of Heuristic Approaches: We conduct a structured and up-to-date review of prominent heuristic-based workload scheduling algorithms—including Min-Min, Max-Min, HEFT, Greedy Resource-Aware Heuristics, Tabu Search, and others—that have been applied in the context of hybrid edge–cloud systems. Each algorithm is examined in terms of its working principles, strengths, limitations, and targeted scheduling objectives.
- Design of a Weighted Scoring Model (WSM): We propose a multi-criteria decision-making framework that evaluates heuristic algorithms using a Weighted Scoring Model. The model is based on seven critical evaluation dimensions: Latency Awareness, Energy Efficiency, Adaptability, Scalability, Scheduling Accuracy, Implementation Complexity, and Resource utilization.
- Comparative Analysis and Ranking: By applying the WSM to a curated set of heuristic algorithms, we provide a transparent and reproducible comparison. Each algorithm is scored and ranked, highlighting trade-offs and performance gaps. This offers valuable guidance for researchers and system architects seeking to select or customize scheduling strategies for specific use cases.

The rest of the paper is organized as follows: Section II reviews the related work on heuristic-based workload scheduling in edge–cloud environments. Section III provides background information and foundational concepts. Section IV introduces a classification of heuristic scheduling approaches. Section V presents representative heuristic algorithms commonly used in edge-cloud systems. Section VI details the methodology of the Weighted Scoring Model used for evaluation. Section VII compares the selected heuristic algorithms based on defined criteria. Section VIII discusses the results of the comparison and highlights key insights. Section IX outlines current challenges and suggests future research directions. Finally, Section X concludes the paper.

II. Related Work

Several recent systematic reviews have explored task scheduling challenges in edge–cloud environments, particularly through the lens of heuristic and metaheuristic techniques. For example, the study in [5] introduced a hybrid scheduling approach that enhances traditional heuristics with deep learning by combining Particle Swarm Optimization (PSO) and the Firefly Algorithm. This fusion leveraged the complementary strengths of both methods and achieved improved scheduling performance for resource-intensive tasks in cloud platforms.

In another work [6], the authors tackled the complexity of scheduling IoT applications in fog computing—a problem recognized as NP-hard. While the paper acknowledges the historical reliance on rule-based and metaheuristic techniques, it also critiques their dependence on global knowledge and limited adaptability in dynamic environments. To address these limitations, they proposed DRLIS, a Deep Reinforcement Learning-based scheduler, and demonstrated its superiority over classical metaheuristics such as NSGA2 and NSGA3, particularly in terms of convergence speed, scheduling time, and optimization effectiveness.

Complementing these algorithmic proposals, the review presented in [4] examined the advantages, limitations, and application contexts of various heuristic-based task scheduling methods within cloud computing. Such analyses are valuable for informing the selection of heuristics in real-world task scheduling scenarios.

Further, [7] offered a broad survey of optimization-driven scheduling algorithms in edge–cloud systems, highlighting techniques such as Genetic Algorithms (GA), Cat Swarm Optimization, and multi-objective variants of Ant Colony Optimization (ACO) and PSO. Their work emphasized the role of multi-objective optimization in balancing trade-offs among latency, energy efficiency, cost, and load distribution. The

study also shed light on emerging themes like adaptive scheduling, collaborative resource orchestration, and addressed open challenges such as handling task dependencies, dynamic resource availability, and fault tolerance—key issues in evolving edge–cloud scenarios.

The study in [8] provides a systematic review of heuristic-based scheduling methods for IoT-fog environments. It classifies approaches into six categories—priority-based, greedy, metaheuristic, learning-based, hybrid, and nature-inspired—though labeled as a taxonomy, the structure is more accurately a flat classification. The review follows PRISMA guidelines and identifies key challenges such as heterogeneity, scalability, and security, while suggesting future directions like integrating machine learning and enhancing explainability. However, it lacks cross-category performance comparisons, broader contextualization with non-heuristic or AI-based methods, and deeper analysis of real-world applicability and security-aware heuristics. Despite these gaps, it remains a valuable reference for heuristic scheduling in fog-cloud IoT systems.

Despite such valuable contributions, there continues to be a gap in the literature: there has not yet been a review dedicated entirely to heuristic-based scheduling of workloads in the special case of hybrid edge–cloud computing. This paper seeks to address the imbalance by providing a comprehensive investigation and taxonomy of heuristic methods, comparing their efficiency, and evaluating their usability in practice in the environments of edge-clouds.

III. Background

A. Edge-Cloud Architecture Overview

Edge-cloud computing integrates two mutually complementary levels of computing infrastructure: edge nodes, located near devices producing data such as sensors and IoT devices, and remote cloud servers, providing powerful computing and massive storage capacity. The architecture develops a hierarchy of computing such that tasks could be optimally redistributed over multiple sites depending upon their latency sensitivity as well as resource requirements. Tasks with high response requirements or those of high volume of data requiring processing over real-time are usually served at the edge, such that the latency for the purpose of intercommunication could be reduced as well as bandwidth usage. Compute-intensive tasks and high-duration data storage tasks are, on the other hand, offloaded to clouds where higher-end resources. Balancing the workload judiciously but adequately between both the edge as well as the cloud layers of such architecture permits providing greater system responsiveness as well as lower communications overhead between distributed applications, but particularly greater scalability as well as fault tolerance of the application [9], [10].

B. Workload Characteristics

Edge–cloud computing workloads are extremely variable and very dynamic, arising from an enormously large variety of sources like sensors, mobile phones, camera-equipped phones, and machines. They very notably fluctuate with computing intensity, latent needs, volume of data, as well as arrival rates [11]. Some require instant processing immediately with no wait time involved, while some have bulk or occasional data. Making the matter worse in such an instance is the changing and diverse nature of the arrival of tasks. At any instant, workloads peak and the system has to decide on the spur of the moment, that is, at real-time scheduling. Schedulers are compelled persistently to shift configurations such that good performance as well as reliability are sustained, such factors as availability of computing resources, state of the current network, as well as priority or requirement of arriving tasks are being considered [11]. Service-level objective (SLO) realization within such an environment requires intelligent mechanisms of scheduling that could adequately contend with ever-changing demands.

C. Scheduling Objectives

Scheduling edge–cloud computing workload is specifically challenging because there exist multiple, often competing objectives that have to be optimized. Efficient schedulers must reconcile multiple desirable objectives in balancing for system efficiency as well as application performance:

- Latency Reduction: Most edge computing initiatives revolve around reducing response time. Scheduling policies must prefer deploying the work next to where data is being generated [12]; lower round-trip delay, and best-case real-time response.
- Enhancing Energy Efficiency: The edge nodes will frequently be deployed at sites that have minimal access to power sources, e.g. at remote sites or at battery-based deployments. Overloading such nodes may lead to energy drain and interruption of service [13]. Scheduling algorithms will then need to target the reduction of transmission of useless information and overloading by nodes with energy limitations.
- Sustaining Load Balancing: Non-balanced allocation of tasks could end up overloading some nodes while leaving other nodes idle. Schedulers will have to avoid skewed allocation of workloads in order to sustain system stability as well as system throughput by avoiding bottlenecks and allowing for more balanced program execution across the network [11].
- Ensuring Quality of Service (QoS) and Quality of Experience (QoE): Some applications of the real-time, i.e. video streaming, software for industrial control or healthcare monitoring, require hard guarantees regarding their performance. Schedulers are thus provided Service-Level Agreements (SLAs) by offering time completion of tasks reliably and correctly [11]. That affects the user experience of service quality directly and thus forms a crucial consideration of current scheduling schemes.

Briefly speaking, the optimal scheduler in edge-cloud needs to be multi-objective; that is, it is required to serve latency, energy, load distribution, and service quality at the same time within a resource-limited and varying environment.

D. Limitations of Traditional Scheduling Approaches

Traditional scheduling methods applied for cloud computing, that are usually based on static policy or

algorithmic deterministic methods, do not perform well within edge–cloud environment scenarios. Traditional methods are usually not dynamic and thus inefficient when addressing real-time variation of resource availability and workload. They are also usually coupled with high computing overhead and are not suitable for systems that lack computing power [14].

Scalability is also a concern. As larger edge networks of thousands of nodes and devices are needed, the majority of typical schedulers do not scale or perform well. Furthermore, most of them rely upon centralized decisions and consequently yield greater incidences of point-of-failure occurrence and communication latency and lower fault tolerance [15]; each of these is a serious weakness of the distributed, heterogeneous environment.

These limitations impose an immediate necessity for lightweight scheduling mechanisms that are also decentralized and adaptive. Heuristic-based mechanisms of these types have demonstrated extraordinary potential. Owing to their tendency to give fast approximate results with low overhead that automatically adjust with changing situations [13], [16], they are very fitting to the new edge–cloud computing requirements.

IV. Classification of Heuristic Scheduling Approaches

Heuristic scheduling algorithms are essential for dealing with the dynamic and resource-constrained nature of edge–cloud computing workloads. Heuristic algorithms are valued for giving good, near-optimum solutions with little overhead, which is a highly desirable characteristic in highly dynamic environments. As indicated in Fig. 1, we put forward a taxonomy of heuristic methods of scheduling, which categorizes the heuristics into simple heuristics, metaheuristics, and hybrid heuristics, based on their underlying strategies and level of complexity [17], [18], [19], [20], [21].

A. Simple Heuristics

Simple heuristics are rule-based scheduling algorithms that use simple, predetermined logic to make decisions. They are most popular in real-time systems because of their low complexity and fast execution, which makes them suitable for applications in situations where fast, lightweight decision-making is critical [22]. These techniques, though not necessarily optimal in terms of system-wide performance, ensure consistent baseline behavior with low processing overhead. Common implementations of simple heuristics include:

- First-Come, First-Served (FCFS): FCFS puts tasks in a strict sequence of submission order without considering the availability of resources and the priority of the task. Even if FCFS is extremely simple in implementation, its performance under dynamic workload is typically poor, which results in under-utilization of resources, increased waiting times, and spikes in latency [23], [24].
- Shortest Job First (SJF): Runs jobs having the shortest estimated running time, trying to reduce average total wait time [23], [24]. Estimation of running time of the tasks may get complex in an edge-cloud scenario, which restricts the usage of SJF despite its theoretical efficiency.
- Min-Min and Max-Min: These algorithms schedule tasks based on their expected completion times. Min-Min selects tasks with the shortest completion time, favoring smaller tasks and aiming to quickly reduce the number of pending jobs. Max-Min, on the other hand, gives priority to tasks with the longest completion time, helping to balance load and avoid long-tail delays [22]. Both are commonly used in cloud systems to enhance throughput and resource efficiency.
- Round Robin (RR): Distributes CPU time equally among all tasks using a time-slicing approach. While RR is fair and prevents starvation, it can incur high context-switching overhead, especially when dealing with a large number of short-lived or high-priority tasks [22], [23].

Overall, simple heuristics offer a solid foundation for scheduling in resource-constrained edge-cloud environments, particularly when quick, interpretable decisions are preferred over optimality.

B. Metaheuristics

Metaheuristic algorithms are advanced, high-level strategies designed to efficiently explore large and complex solution spaces—making them well-suited for challenging optimization tasks such as workload scheduling in edge–cloud environments. Unlike simple heuristics that rely on fixed rules, metaheuristics employ adaptive mechanisms to balance exploration and exploitation, allowing them to find near-optimal solutions even in highly dynamic and multi-objective settings.

Some of the most widely used metaheuristic approaches include:

- Genetic Algorithm (GA): Inspired by the principles of natural selection and genetics, GAs use operators like selection, crossover, and mutation to evolve a population of candidate solutions across generations. GAs are particularly effective at navigating large and nonlinear problem spaces, but they often require significant computational time and tuning, especially in real-time scenarios [25], [26].
- Particle Swarm Optimization (PSO): This algorithm models a group of agents (particles) moving through the solution space, each adjusting its position based on its own experience and the performance of neighboring particles. PSO is known for its simplicity, fast convergence, and low memory footprint, which makes it appealing for real-time task scheduling in distributed systems [26], [27].
- Ant Colony Optimization (ACO): Inspired by the way real ants find optimal paths to food, ACO simulates a colony of artificial ants that build solutions incrementally, guided by pheromone trails. ACO is well-suited for discrete and combinatorial optimization

problems, such as scheduling tasks with dependencies or constraints [28].

- Simulated Annealing (SA): Modeled after the metallurgical process of annealing, SA probabilistically accepts worse solutions at early stages to avoid becoming trapped in local minima. This ability to escape suboptimal solutions makes SA effective for solving complex scheduling problems where traditional methods may get stuck [26], [27], [29].

Recent research has demonstrated the effectiveness of metaheuristics in edge–cloud systems. For example, ACO-based schedulers have been used to optimize multiple objectives—such as latency, energy consumption, and cost—through fitness functions tailored to dynamic edge environments. Another study proposed a modified Firefly Algorithm for workflow scheduling in hybrid cloud–edge platforms, which led to noticeable improvements in makespan and execution cost [28], [30].

Metaheuristics, while more resource-intensive than simple heuristics, offer flexibility and robustness in finding high-quality solutions for multi-objective and time-sensitive scheduling problems—a critical need in modern edge–cloud systems.

C. Hybrid Heuristics

Hybrid heuristics are scheduling strategies that combine multiple heuristic or metaheuristic methods in order to leverage their individual strengths while compensating for their inherent weaknesses. By integrating different approaches—such as combining the fast convergence of one algorithm with the global search capability of another—hybrid methods aim to achieve more balanced and effective solutions, particularly for complex scheduling problems in edge–cloud environments [8].

These strategies have shown strong potential in optimizing multiple, often conflicting objectives such as execution time (makespan), energy consumption, and operational cost. For instance, one study proposed a hybrid metaheuristic algorithm to enhance IoT service placement in edge–cloud infrastructures, achieving improved performance across key scheduling metrics [26].

Although hybrid approaches are not inherently AI-based, they still fall within the broader category of heuristic-driven scheduling techniques. However, when such methods are further enhanced with machine learning models or neural components, they begin to evolve into AI-driven schedulers—offering even greater adaptability and decision-making intelligence in dynamic environments.

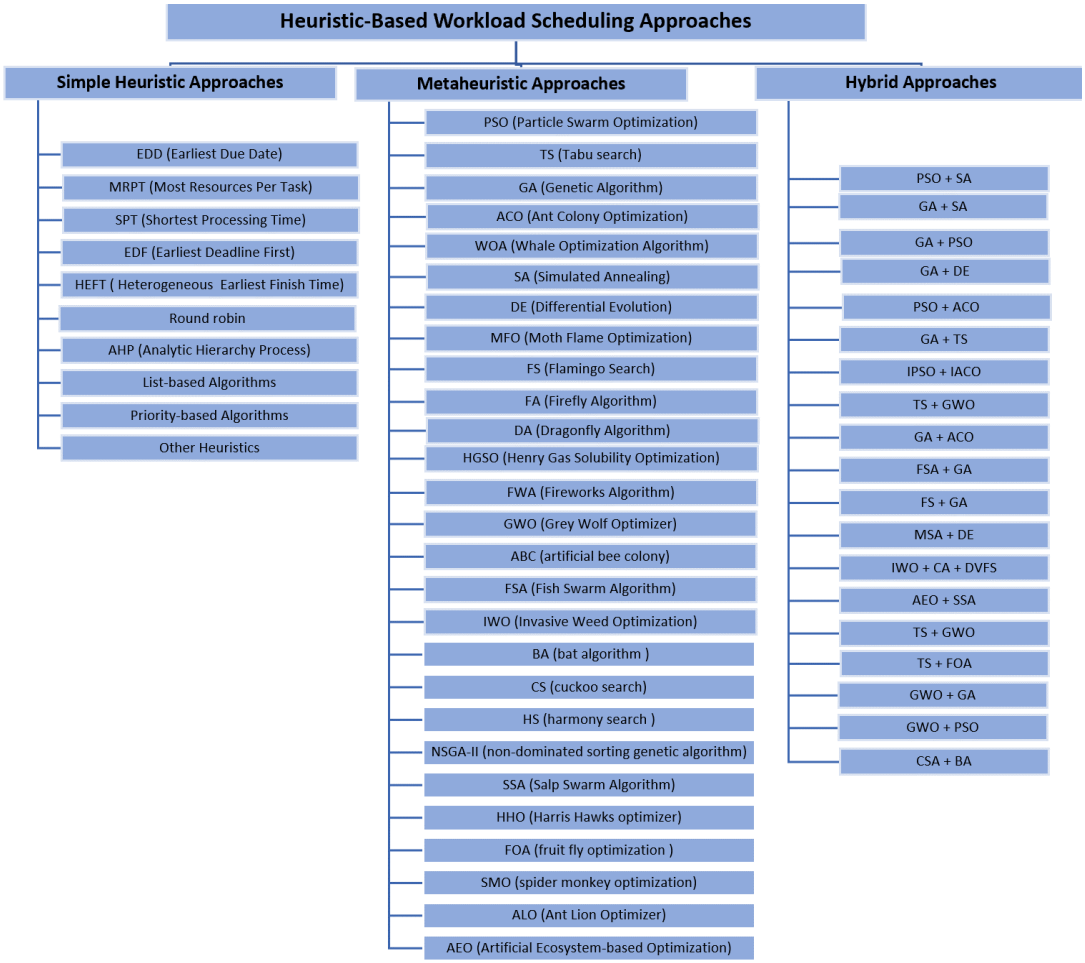

Fig. 1. Taxonomy of heuristic-based workload scheduling approaches.

V. Representative Heuristic Approaches

We present in this section some representative heuristic and metaheuristic approaches that have been used for edge-cloud system workload scheduling. These approaches take optimal task-to-resource allocations under latency, energy, and workload balancing constraints into consideration.

This paper chooses seven established scheduling algorithms: Greedy Resource-Aware Heuristics, HEFT, Min-Min/Max-Min, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Tabu Search for comparative analysis. The choice was based on several objective criteria aiming to ensure relevance and diversity. Initially, the chosen algorithms exhibit a variety of methodological approaches: rule-based (HEFT, Min-Min/Max-Min), greedy (Greedy Resource-Aware), evolutionary (GA, PSO), swarm intelligence (ACO), and memory-based search (Tabu Search). Such variety provides balanced coverage of diverse scheduling paradigms. Second, these methods have appeared extensively in edge–cloud literature and in reputable journals, establishing their research relevance. Third, they have exhibited performance in dealing with fundamental edge–cloud issues such as sensitivity to latency, limited resources, and dynamic workloads. They have been popularly instantiated in simulation environments such as CloudSim, iFogSim, and EdgeCloudSim. Fourth, they can be obtained in the form of open-source code or can be easily coded, thereby ensuring reproducibility and fair comparison in a unified Weighted Scoring Model (WSM) environment. Lastly, the combination of the above factors makes the selected set of algorithms both pragmatically valuable and representative of the general landscape of heuristic-based edge–cloud scheduling.

A. Heterogeneous Earliest Finish Time (HEFT)

The HEFT algorithm is a well-known static heuristic for scheduling in heterogeneous computing environments. It assigns tasks based on upward rank and maps each task to the processor that incurs the minimum earliest finish time. HEFT has been popular in edge–cloud environments because of its efficiency in reducing overall makespan and its ease of implementation [22], [31]. The major characteristic of HEFT is to consider the earliest finish time, though it does not allow rescheduling in the event of dynamic change [32] and does not account for energy consumption, which proves to be a main drawback in edge environments [33]. In terms of adaptability, HEFT works with static DAGs (Directed Acyclic Graph) and pre-estimated task times, lacking dynamic feedback or re-optimization capabilities. It performs well in controlled environments but is not responsive to changes. Regarding resource utilization, HEFT uses task execution time and communication cost to prioritize task-node mapping, which helps achieve efficient load balancing and reduce idle time [34]. Minimizing the earliest finish time enables efficient resource distribution across heterogeneous nodes, making it well-suited for static resource planning.

B. Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) is a metaheuristic inspired by the foraging behavior of ants, where virtual “ants” probabilistically explore task assignments based on pheromone trails that reflect the quality of previous solutions. ACO has shown promise in optimizing multi-objective scheduling in dynamic edge–cloud environments by adapting to changing network or resource conditions [30]. Its adaptability stems from the use of pheromone updates, which encode historical experience and influence future scheduling decisions. However, ACO may require multiple iterations to converge, making it less responsive to immediate changes [35]. In terms of resource utilization, ACO builds better task-to-resource mappings over time, often leading to improved resource use as pheromone trails reinforce successful assignments. That said, its early iterations may result in suboptimal utilization before convergence is achieved [36]. ACO also faces several limitations: in its basic form, it is suitable only for small problem instances like the Traveling Salesman Problem with a limited number of cities, and it exhibits relatively high memory complexity of $O(n^2)$ [37]. Moreover, it is known to suffer from parameter sensitivity, limited scalability to large-scale problems, and challenges in real-time implementation [38].

C. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) poses task assignment as a problem of swarm intelligence, where particles representing candidate solutions explore the search space based on both individual and collective experience [26]. PSO has been successfully applied in edge–cloud systems for task scheduling under latency and energy constraints, particularly in large-scale or uncertain workload scenarios [22]. Its popularity stems from its simplicity and effectiveness in handling nonlinear, multi-modal optimization problems, as well as its adaptability to various contexts such as scheduling, resource allocation, and feature selection [33]. From an adaptability viewpoint, PSO changes task placement of particles across iterations, gradually responding to changing constraints, albeit as a delayed responder and potentially requiring several iterations to adapt well to change [39]. From a resource utilization viewpoint, PSO changes workloads across resources based on the resources available, simulating particles converging to optimum node locations. The nature of this approach prevents skewed resource utilization and minimizes underutilization, which makes PSO a popular approach to resource-aware scheduling [39]. The computational cost, however, depends on the number of particles, the dimensionality of the solution space, and the iterations, and has a general $O(n \times d \times t)$ time complexity [40].

D. Min-Min and Max-Min Heuristics

Min-Min and Max-Min are popular but straightforward list schedule heuristics employed in heterogeneous computing environments. Min-Min chooses the task possessing the minimum of the earliest completion time on any available resource, and it performs well in cases where smaller tasks become prevalent, as they incur less waiting time and get processed faster [41], [42]. It has, however, a major flaw of causing large tasks to be delayed heavily, as they usually get pushed aside in favor of smaller ones [42]. Max-Min, on the other hand, chooses the task possessing the maximum of the minimum completion times and ends up assigning bigger tasks to the best node to complete them in the shortest time. It prevents large tasks from remaining perpetually on hold and suits well in cases of heterogeneous environments having varied requirements of the tasks and varying node capabilities.

There, however, exists the possibility of Max-Min getting inefficient in cases of high prevalence of tiny tasks, as they incur extra and needless waiting time [42]. Both of the above algorithms are deterministic and static, in that they adhere to fixed priority assignment policies and do not respond to varying levels of task arrivals, system loading, and availability of the nodes. Consequently, they are not suitable in dynamic or mobile edge–cloud environments [22], [42]. That implies, there exists no re-evaluation of the tasks once they become scheduled, and hence, they become inflexible in fluctuating conditions. In terms of resource utilization efficiency, they end up overloading faster nodes and keeping other ones idle, as a result of their static selection of the task and failure to adapt to the runtime variations of resource status.

E. Greedy Resource-Aware Heuristics

Greedy Resource-Aware Heuristics (GRAH) are high-speed, lightweight, and context-aware heuristics of scheduling, making locally optimal decisions based on real system parameters such as CPU utilization, memory, bandwidth, and latency to edge or cloud nodes [12], [43]. They become particularly valuable in real-time and latency-sensitive environments, in which they prioritize task assignment to the nearest or least-loaded node with adequate resources, in order to keep data transfer latency to a minimum and increase responsiveness. GRAH approaches have been widely implemented in mobile and fog computing scenarios due to their simplicity, low overhead, and adaptability to runtime conditions [42]. Unlike complex global optimizers, greedy algorithms do not require prior knowledge of future workloads, making them well-suited for dynamic edge–cloud environments. Their use of the current system state enables on-the-fly scheduling, ideal for large-scale, time-critical systems. In terms of adaptability, GRAH performs well by continuously reacting to changes in system load and node availability. Regarding resource utilization, these heuristics respond quickly to imbalances (e.g. by assigning tasks to the least-loaded nodes), but they lack a global view of the system. This can result in uneven distribution or suboptimal long-term resource use [44]. While GRAHs are generally more energy-efficient and responsive in fog or edge environments, they may struggle to achieve globally optimal performance when system dynamics evolve over time or require coordinated optimization [21], [42].

F. Tabu Search

Tabu Search (TS) is a metaheuristic optimization algorithm for solving complex combinatorial problems such as task scheduling, routing, and resource placement. It excels in large search spaces where traditional heuristics could become entrapped in local optima [45], [46]. In edge–cloud workload scheduling, TS may be applied to refine task assignments and investigate neighboring solutions near an initial schedule or one obtained heuristically. A hallmark of Tabu Search is the addition of a tabu list, a memory-oriented mechanism that prevents the algorithm from revisiting recently explored or suboptimal solutions. This encourages wider exploration and helps avoid cycles [46], [47]. TS can be easily extended to multi-objective optimization, effectively balancing trade-offs like makespan versus cost and load versus latency [45], [46]. TS excels in heterogeneous and dynamic environments, where its memory-guided refinement process helps escape local optima and reassign tasks based on system changes, demonstrating high adaptability [46], [48]. From a resource utilization standpoint, TS outperforms simpler heuristics by continuously refining how workloads are distributed. However, its success depends on proper configuration and how well it converges [46]. By evaluating neighboring solutions while avoiding previously visited ones, TS can maintain better overall resource balance across edge and cloud nodes.

G. Genetic Algorithm

Genetic Algorithm (GA) is an evolutionary optimization algorithm based on the natural selection process that works by evolving a group of possible solutions using operations like selection, crossover, and mutation. Over successive generations, these steps help improve how tasks are scheduled. In edge–cloud workload scheduling, GA is often used for optimizing multiple objectives at once, such as reducing makespan, energy consumption, resource usage, and execution cost, especially in heterogeneous and large-scale environments where exhaustive search becomes impractical [49]. GA is especially valued for its ability to discover near-optimal solutions in difficult, constraint-heavy problems, and it allows fine-tuning based on different factors like latency, energy, and cost. Many GA-based systems use customized fitness functions that specifically target better resource utilization and load balancing, which improves throughput and makes better use of CPU and memory [49], [50]. Still, implementing GA can be quite complex. It requires careful planning for parameters like population size, rates for crossover and mutation, and how fitness is calculated [49], [51]. While GA can adapt to changing requirements by gradually evolving its solutions, it isn't known for quick responsiveness since several generations are usually needed to adjust to changes [39]. Nonetheless, GA's strong scalability and solid performance across varied workloads make it a robust, albeit resource-intensive, solution for hybrid scheduling systems [49], [50].

These representative heuristic approaches represent the diversity and effectiveness of the strategies used for task scheduling optimization for edge-cloud environments. All the approaches exhibit their own strengths and weaknesses, contributing to the ongoing improvement of effective scheduling solutions.

VI. Weighted Scoring Model

Weighted Scoring Model (WSM) is among the methods of comparison as well as decision-making between options described by predefined criteria. The process is frequently applied in order to compare different algorithms.

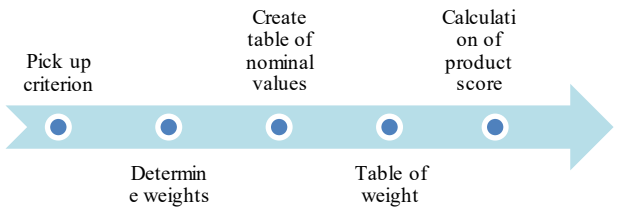

Fig. 2. WSM process [52].

Application of the method often happens through a series of steps, such as in Fig. 2 [52]:

- Step 1: Establish the criteria to be employed for each option.
- Step 2: Assign weights for the criteria as percentages of their priority.
- Step 3: Create a table summarizing nominal values for all of the criteria across different options.
- Step 4: Create a table displaying weighted values for all criteria with weights as percentages, ensuring the total weight sums to 100%.
- Step 5: Calculate weighted scores by multiplying nominal values by weights for each option [52].

VII. Comparison of Heuristic Algorithms

This section presents a comparative analysis of illustrative heuristic-based workload scheduling strategies for edge-cloud infrastructure. A comparison of these algorithms is done on the basis of some of the most significant factors of edge-cloud deployment [22], [53], such as latency awareness, energy efficiency, scalability, adaptability, Scheduling Accuracy, Implementation Complexity, and Resource Utilization, as mentioned in Table I.

A. Application of the Weighted Scoring Model

Weighted Scoring Model evaluates heuristics-driven workload scheduling algorithms on a variety of critical factors, each with a specific weight based on its relative importance in edge–cloud scenarios, as shown in Table II. Latency Awareness and Adaptability each have a weighting of 20%, since they serve a prime function in offering minimum delay and responsiveness in highly dynamic environments with changing workloads and mobility. Energy Efficiency and Scalability each have a weighting of 15% due to their function in enabling sustainable operations in resource-constrained edge devices and expanding infrastructure. Scheduling Accuracy, Implementation Complexity, and Resource Utilization each have a weighting of 10%, in consideration of their function in offering pragmatic, accurate, and cost-efficient task placement, but they can be impacted indirectly by factors of greater priority. This distribution strikes a balance between performance-critical metrics (latency, adaptability) and operational concerns (energy, complexity, utilization) tailored to hybrid edge–cloud contexts.

This weighted method allows for a systematic and clear comparison of the heuristic techniques, to support the choosing of the most suitable schedule strategy in hybrid edge–cloud. Every algorithm is rated on a score of 1 to 5 (n/a = 0), as in Table III, and multiplied by its corresponding weight. Table IV contains the final weighted scores.

It should be noted that the weighting scheme, while grounded in trends consistently emphasized in the edge–cloud scheduling literature, inevitably involves a degree of subjectivity. To mitigate this, we relied on recurring priorities reported across multiple surveys and empirical studies (e.g. the centrality of latency and adaptability in real-time applications). The influence of these parameters is that absolute rankings may vary under alternative weighting choices; however, the relative positioning of algorithms along specific criteria (e.g. some excelling in latency but weaker in energy efficiency) remains consistent with findings reported in the literature. The presented WSM is therefore intended as a structured synthesis tool rather than an absolute ranking, providing practitioners with a transparent framework that can be adapted or refined with alternative weighting schemes or sensitivity analyses in future studies.

It should be noted that the weighting scheme, while grounded in trends consistently emphasized in the edge–cloud scheduling literature, inevitably involves a degree of subjectivity. The influence of these parameters is that absolute scores or rankings may shift under alternative weighting schemes; however, the relative strengths and weaknesses observed for each algorithm (e.g. HEFT excelling in scheduling accuracy but lacking adaptability, or Greedy heuristics performing well in responsiveness) remain consistent with empirical evidence reported across prior studies. The presented WSM is therefore intended as a structured synthesis tool rather than an absolute benchmark. Future research could extend this work by conducting sensitivity analyses on the weighting distribution or by employing alternative multi-criteria decision-making frameworks such as AHP, TOPSIS, or entropy-based weighting to further validate robustness.

B. Comparison

Fig. 3 illustrates the comparative performance of seven scheduling algorithms across seven evaluation criteria: Latency Awareness, Energy Efficiency, Scalability, Scheduling Accuracy, Implementation Complexity, Resource Utilization, and Adaptability.

TABLE I. Comparison Criteria

Criterion	Description
Latency Awareness	Ability to minimize task response time and meet real-time constraints [53].
Energy Efficiency	Capacity to optimize energy usage, especially at resource-constrained edge nodes [41].
Scalability	Ability to perform well as the number of tasks/nodes increases [53].
Adaptability	Flexibility to operate under dynamic workloads and network conditions [53].
Scheduling Accuracy	measures how effectively an algorithm assigns tasks to appropriate resources, matches task needs with node capabilities, prevents overload or failures, and meets quality of service (QoS) goals like latency and availability.
Implementation Complexity	effort and overhead required to develop, deploy, and maintain a scheduling algorithm, considering factors such as algorithmic simplicity, computational requirements, integration effort, and tuning difficulty.
Resource Utilization	measures how efficiently a scheduling algorithm leverages available computing, memory, storage, and network resources, aiming to maximize usage while avoiding both underutilization and overload.

TABLE II. ASSOCIATED WEIGHT TO EACH CRITERION

Criterion	Abbreviation	Proposed weight
Latency Awareness	LA	20%
Energy Efficiency	EE	15%
Scalability	S	15%
Scheduling Accuracy	SA	10%
Implementation Complexity	IC	10%
Resource Utilization	RU	10%
Adaptability	A	20%

TABLE III. COMPARISON OF SCHEDULING ALGORITHMS IN EDGE–CLOUD ENVIRONMENTS

Algorithm	LA	EE	S	SA	IC	RU	A
Greedy Resource-Aware	5	4	5	3	4	3	5
Tabu Search	4	3	3	4	2	3	4
ACO	4	4	3	4	2	3	3
PSO	3	3.5	4	3	3	4	3
GA	3	3	4	4	2	4	3
HEFT	3	2	3	5	3	4	2
Min-Min / Max-Min	2	2	3	3	5	3	1

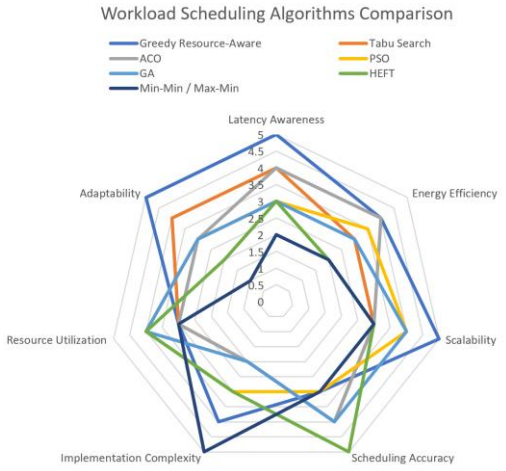

Fig. 3. Workload scheduling algorithms comparison across key evaluation criteria.

TABLE IV. WSM RESULTS

Algorithm	WSM Score	WSM Score (%)
Greedy Resource-Aware	4.35	87
Tabu Search	3.4	68
ACO	3.35	67
PSO	3.33	66.6
GA	3.25	65
HEFT	2.95	59
Min-Min / Max-Min	2.45	49

The Greedy Resource-Aware algorithm shows consistently high values in Latency Awareness, Scalability, and Adaptability, with moderately high scores in Energy Efficiency and Implementation Complexity. Its performance is slightly lower in Scheduling Accuracy and Resource Utilization.

Tabu Search demonstrates relatively high values in Latency Awareness, Scheduling Accuracy, and Adaptability, while showing moderate performance in Energy Efficiency and Resource Utilization. It records a lower score in Implementation Complexity.

ACO (Ant Colony Optimization) displays balanced scores across most criteria. It performs moderately well in Latency Awareness, Energy Efficiency, and Scheduling Accuracy, with slightly lower values in Adaptability and Implementation Complexity.

Genetic Algorithm (GA) shows strong performance in Scalability, Scheduling Accuracy, and Resource Utilization, with slightly lower values in Latency Awareness, Energy Efficiency, and Adaptability. It scores low in Implementation Complexity.

Particle Swarm Optimization (PSO) scores highest in Resource Utilization, with good values in Scalability and Energy Efficiency. It has moderate values in the remaining criteria, except for Latency Awareness and Adaptability, which are slightly lower.

HEFT shows its strongest score in Scheduling Accuracy, followed by relatively good performance in Implementation Complexity and Resource Utilization. It records lower values in Energy Efficiency, Adaptability, and Scalability.

Min-Min / Max-Min scores highest in Implementation Complexity, followed by moderate values in Scheduling Accuracy and Resource Utilization. It records lower scores in Latency Awareness, Energy Efficiency, Adaptability, and Scalability.

VIII. RESULTS AND DISCUSSION

Our comparison reveals clear differences in how each scheduling algorithm performs under various conditions. Greedy Resource-Aware Heuristics came out strong, particularly in latency awareness, scalability, and adaptability. Their strength lies in their simplicity and speed—they make quick decisions based on current system conditions, which is ideal for fast-changing edge–cloud environments. While they don't always find the most globally optimal solution, their ability to respond instantly makes them very practical for real-time applications.

Tabu Search also performed well, striking a good balance between accuracy and adaptability. Its use of memory to avoid repeating poor decisions allows it to improve over time and adjust to changing workloads. ACO and PSO showed solid results, especially in energy efficiency and resource usage. However, they typically need more fine-tuning and take longer to settle on the best solution. Genetic Algorithms offered strong scalability and were effective at handling multiple objectives, but their complexity and tuning requirements may pose challenges in real deployments. By comparison, HEFT

excelled in schedule predictability due to its rigorous methodology, but it is less adaptable to varying scenarios since it is inflexible. Min-Min and Max-Min, while they are easy to program, have limited flexibility and weaker performance in scenarios with sensitive latency constraints.

From a quantitative perspective, the average WSM score across all algorithms was 3.29 (65.8%), with values ranging from 2.45 (49%) for Min-Min/Max-Min to 4.35 (87%) for Greedy Resource-Aware. This 38-point spread underscores the heterogeneity of heuristic methods. Greedy Resource-Aware outperformed the lowest-scoring algorithm by 33%, reflecting its superior trade-offs across latency, scalability, and adaptability. Fig. 4 illustrates these differences through the final WSM scores, making clear the ranking and relative gaps among the seven algorithms.

At the criterion level, Scalability achieved the highest average score (3.57), while Adaptability was the lowest (3.0), confirming the literature's emphasis on adaptability as a persistent challenge. Pairwise comparisons highlight specific trade-offs: HEFT delivered the best Scheduling Accuracy (5.0), which was 66% higher than Greedy Resource-Aware (3.0), but performed poorly in adaptability (2.0). Conversely, PSO and GA achieved the strongest Resource Utilization (4.0), surpassing Greedy Resource-Aware (3.0) by 33%. Greedy Resource-Aware, however, exceeded GA in Adaptability by 29% (5 vs. 3). These trade-offs are further illustrated in Fig. 5, which shows the normalized contribution of each evaluation criterion to the overall performance of each algorithm, making visible the strengths and weaknesses across dimensions.

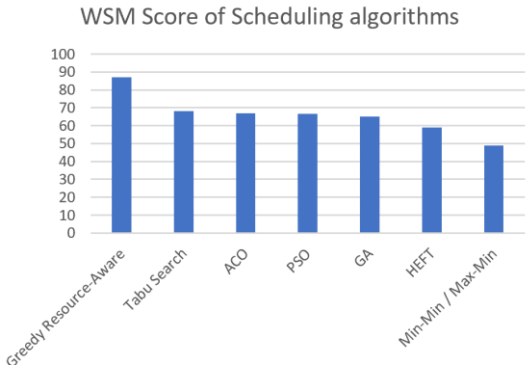

Fig. 4. WSM scores of scheduling algorithms.

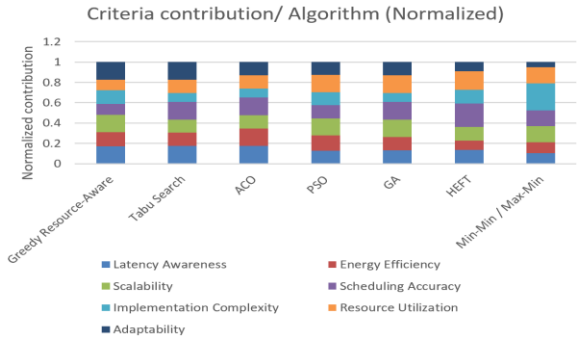

Fig. 5. Normalized contribution of each evaluation criterion to the overall performance of each algorithm.

Overall, our findings emphasize that adaptability is a major strength of edge–cloud scheduling. Algorithms, such as Greedy Heuristics and Tabu Search that can adapt on the fly do better in real-world scenarios. Simpler does not have to be weaker; Greedy approaches are excellent where quick decisions are more significant than optimal optimization. However, where accuracy or balancing several objectives matters more, more complex approaches such as ACO or Tabu Search may be more suitable. These findings provide useful direction to system designers in choosing the best approach suitable for their application's requirements.

It is important to note, however, that the comparative analysis presented here is derived from a literature-driven synthesis using the Weighted Scoring Model rather than from direct experimentation on large-scale, real-world edge–cloud testbeds. While this approach ensures a structured and transparent comparison across algorithms, it does not fully capture deployment complexities such as heterogeneous hardware, dynamic user mobility, or large-scale resource contention. The results should therefore be interpreted as a framework to guide algorithm selection rather than as definitive performance benchmarks. Future work could strengthen these findings by validating them through empirical studies on realistic testbeds or by performing sensitivity analysis on the weighting scheme to assess its influence on the rankings.

In addition, while prior works have evaluated individual heuristics or small subsets of algorithms, only a limited number of studies have attempted unified multi-criteria comparisons. For instance, some surveys assess algorithms qualitatively or focus narrowly on metrics such as energy efficiency or latency, but they do not provide a structured weighting and scoring process. Our WSM-based framework complements these efforts by offering a transparent, quantitative approach that integrates multiple evaluation dimensions. Nonetheless, a broader comparison with other decision-making frameworks and benchmark studies remains an important direction for future work.

IX. Challenges and Future Directions

While heuristic-based scheduling methods for edge-cloud computing have realized significant advances, they still face a series of challenges. Such challenges present tenable research opportunities for the enhancement of adaptability, efficiency, and scalability in the management of workloads on hybrid and distributed infrastructure.

A. Key Challenges

- Limited adaptability in dynamic environments: Many classical heuristics, like HEFT, Min-Min, and Max-Min, are designed for static or semi-static systems. Nevertheless, edge-cloud environments are inherently dynamic, with fluctuating workloads, network conditions, and resource states. Studies have shown that such heuristics can suffer 30–40% degradation in task completion time under dynamic conditions [8], whereas deep reinforcement learning (DRL) schedulers achieve ~25–35% faster scheduling times than metaheuristics

[6]. This gap underscores the need for adaptive or context-aware heuristics that can respond in real time.

- Suboptimal global scheduling caused by local decisions: While greedy heuristics are lightweight and efficient, they often make locally optimal decisions based on immediate node states. However, in large-scale or federated edge-cloud systems, this can result in globally suboptimal outcomes, such as unbalanced load distribution or resource starvation. Prior studies have reported that simple rule-based methods can incur 15–25% longer makespan compared to metaheuristic alternatives [4].
- High computational cost in metaheuristics: Approaches like PSO and ACO offer flexibility and multi-objective optimization. However, their computational demands remain a significant barrier for resource-constrained edge devices. For instance, hybrid PSO–Firefly methods improved resource utilization by ~20% but required 2–3× higher computation time compared to baseline heuristics [5]. This trade-off reduces the feasibility of real-time edge scheduling.
- Scalability in large-scale deployments: While many heuristics perform well in small-scale simulations, efficiency often degrades in large-scale, real-world deployments. Multi-objective GA and PSO variants exhibit scalability but can face exponentially increasing computation times as the number of tasks grows [7]. This scalability bottleneck hinders their application in real-time industrial edge-cloud systems.
- Energy efficiency and sustainability issues: Most heuristics prioritize latency and throughput while overlooking energy constraints. Yet, studies show that energy-aware heuristics can reduce consumption by 10–20% compared to traditional methods [8]. However, battery awareness and energy–latency trade-offs remain largely unaddressed, especially in mobile or IoT contexts.
- Security and privacy concerns: Existing heuristics rarely account for secure task placement. In multi-tenant edge-cloud systems, tasks may involve sensitive data, requiring privacy-preserving or trust-aware scheduling strategies.
- Uncertainty in workload prediction: Many heuristics assume prior knowledge of task execution times or resource availability. In practice, workloads are unpredictable, and inaccurate estimations can lead to degraded performance.
- Lack of standardized benchmarks and validation: Current heuristic evaluations are fragmented across different simulators and testbeds, with no standardized benchmark suite. This inconsistency makes cross-study comparisons difficult and reduces the reliability of results.

B. Future Research Directions

- Designing adaptive heuristics: In future works, focus should be placed on developing context-aware or adaptive heuristics that can deal with changes in workload characteristics, resource availability, and user mobility in real time.
- Lightweight metaheuristic variants: Optimizing and simplifying metaheuristics such as ACO and PSO, through parameter tuning, pruning, or hybridization with greedy rules, can make them feasible for real-time edge scheduling.
- Hybrid scheduling models: It is claimed that adaptive multi-objective schedulers that maintain performance while controlling complexity can be obtained by integrating the different strengths from various heuristics, such as Min-Min combined with PSO or HEFT with greedy prioritization.
- Energy and SLA-aware extensions: adding energy models, battery-awareness, and SLA-awareness to traditional heuristics algorithms could greatly enhance their applicability for edge computing in real life.
- Validation on real-world testbeds and benchmarks: Beyond theoretical and literature-driven comparisons, future studies should validate scheduling strategies through experiments on realistic, large-scale edge-cloud testbeds. Such validation would account for hardware heterogeneity, user mobility, and dynamic network states, providing stronger evidence of practical feasibility. Additionally, integrating standardized benchmarks would allow fairer comparisons across heuristic approaches.
- Comparative frameworks refinement: While the WSM-based evaluation presented in this paper offers a structured multi-criteria perspective, it remains one of the few efforts of its kind. Future work should extend this by comparing WSM with alternative decision-making frameworks (e.g. AHP, TOPSIS, or data-driven weighting methods) to ensure robustness and reduce subjectivity in rankings.

In summary, while heuristic-based scheduling is promising for edge–cloud computing, it faces critical limitations in adaptability, scalability, heterogeneity management, and real-world validation. Quantitative evidence from recent studies highlights performance gaps of up to 40% in dynamic adaptability, 25% in makespan efficiency, and 20% in energy consumption, confirming the need for new strategies. Future research should prioritize the development of lightweight adaptive heuristics, hybrid models, and energy-aware approaches, validated through standardized benchmarks and realistic testbeds. Such efforts will ensure that heuristic scheduling evolves to meet the complex performance demands of next-generation edge–cloud systems.

X. Conclusion

This paper illustrated a comprehensive review and comparative analysis of heuristic-based workload scheduling methods in hybrid edge–cloud systems. Based on the methodological diversity, coverage in literature, and practical applicability, seven representative algorithms: Greedy Resource-Aware Heuristics, HEFT, Min-Min/Max-Min, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Tabu Search were chosen. A Weighted Scoring Model (WSM) was utilized to compare the aforementioned methods on the basis of major criteria: latency awareness, energy efficiency, scalability, scheduling accuracy, implementation complexity, resource utilization, and adaptability.

Results indicate that every algorithm possesses different strengths. Greedy heuristics excel in low-latency and dynamic environments because of their quick responsiveness and reduced complexity. Tabu Search and ACO have good adaptability and multi-objective optimization, but may have long convergence times. PSO has the strength of balance in optimization quality and reduced computational cost, especially in large-scale or uncertain environments. Genetic Algorithms have high scalability and flexibility, but have the cost of increased implementation complexity. Static approaches such as HEFT and Min-Min/Max-Min are still applicable to unstructured or predictable workloads, but they are inflexible in the presence of real-time conditions.

Beyond these descriptive findings, the study highlights several analytical insights. First, there is no universally optimal heuristic: trade-offs between adaptability, scalability, and efficiency remain evident. Second, adaptability consistently emerged as the weakest dimension across algorithms, confirming it as a central research gap in heuristic scheduling. Third, lightweight approaches such as Greedy, while often dismissed as simplistic, may outperform complex methods when responsiveness is prioritized over optimality. These observations provide actionable guidance for selecting scheduling strategies that best align with specific edge–cloud scenarios.

It is important to recognize the limitations of this study. The comparison was restricted to seven algorithms that, while representative, do not capture the full breadth of heuristic and metaheuristic methods. The WSM approach, though systematic, relies on subjectively chosen weights that may bias rankings toward particular criteria. Furthermore, the evaluation was literature-driven and did not involve direct validation on real-world, large-scale edge–cloud deployments. These limitations mean that results should be interpreted as a structured framework for comparative understanding rather than definitive performance benchmarks.

Future work will address these limitations by expanding the algorithm set, experimenting with alternative multi-criteria decision-making frameworks such as AHP or TOPSIS, and validating the findings on realistic testbeds. In addition, further research will explore the design of adaptive and hybrid scheduling methods that integrate the responsiveness of lightweight heuristics with the optimization accuracy of metaheuristics or learning-based approaches. By confronting the challenges of adaptability, heterogeneity, and scalability, such efforts will ensure that heuristic scheduling evolves to meet the complex performance requirements of next-generation edge–cloud systems.

References

[1] M. Goudarzi, S. Ilager, and R. Buyya, “Cloud Computing and Internet of Things: Recent Trends and Directions,” Internet of Things, pp. 3–29, 2022, doi: 10.1007/978-3-031-05528-7_1.

[2] K. P. B. Kanagarla, “Edge Computing and Analytics for IoT Devices: Enhancing Real-Time Decision Making in Smart Environments,” SSRN Electronic Journal, 2024, doi: 10.2139/SSRN.5012466.

[3] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, “Mobile Edge Computing: A Survey,” IEEE Internet Things J, vol. 5, no. 1, pp. 450–465, Feb. 2018, doi: 10.1109/JIOT.2017.2750180.

[4] N. Soltani, B. Soleimani, and B. Barekatain, “Heuristic Algorithms for Task Scheduling in Cloud Computing: A Survey,” International Journal of Computer Network and Information Security, vol. 9, no. 8, pp. 16–22, Aug. 2017, doi: 10.5815/IJCNIS.2017.08.03.

[5] H. Shingne and R. Shriram, “Heuristic deep learning scheduling in cloud for resource-intensive internet of things systems,” Computers and Electrical Engineering, vol. 108, p. 108652, May 2023, doi: 10.1016/J.COMPELECENG.2023.108652.

[6] Z. Wang, M. Goudarzi, M. Gong, and R. Buyya, “Deep Reinforcement Learning-based Scheduling for Optimizing System Load and Response Time in Edge and Fog Computing Environments,” Future Generation Computer Systems, vol. 152, pp. 55–69, Sep. 2023, doi: 10.1016/j.future.2023.10.012.

[7] K. Vinothkumar and D. D. Maruthanayagam, “Issue 3 www.jetir.org (ISSN-2349-5162),” JETIR2203276 Journal of Emerging Technologies and Innovative Research, vol. 9, 2022, Accessed: Jul. 29, 2025. [Online]. Available: www.jetir.org

[8] D. Alsadie, “Advancements in heuristic task scheduling for IoT applications in fog-cloud computing: challenges and prospects,” PeerJ Comput Sci, vol. 10, p. e2128, Jun. 2024, doi: 10.7717/PEERJ-CS.2128/TABLE-8.

[9] H. Nouhas, A. Belangour, and M. Nassar, “Cloud and Edge Computing Architectures: A Survey,” in 2023 IEEE 11th Conference on Systems, Process & Control (ICSPC), IEEE, Dec. 2023, pp. 210–215. doi: 10.1109/ICSPC59664.2023.10420123.

[10] P. Arthurs, L. Gillam, P. Krause, N. Wang, K. Halder, and A. Mouzakitis, “A Taxonomy and Survey of Edge Cloud Computing for Intelligent Transportation Systems and Connected Vehicles,” IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 7, pp. 6206–6221, Jul. 2022, doi: 10.1109/TITS.2021.3084396.

[11] A. A. Ismail, N. E. Khalifa, and R. A. El-Khoribi, “A survey on resource scheduling approaches in multi-access edge computing environment: a deep reinforcement learning study,” Cluster Computing 2024 28:3, vol. 28, no. 3, pp. 1–45, Jan. 2025, doi: 10.1007/S10586-024-04893-7.

[12] [12] Z. Ali et al., “A Comprehensive Utility Function for Resource Allocation in Mobile Edge Computing,” Computers, Materials and Continua, vol. 66, no. 2, pp. 1461–1477, Dec. 2020, doi: 10.32604/cmc.2020.013743.

[13] H. Asghar and E.-S. Jung, “A Survey on Scheduling Techniques in the Edge Cloud: Issues, Challenges and Future Directions,” Feb. 2022, Accessed: Mar. 26, 2025. [Online]. Available: https://arxiv.org/abs/2202.07799v1

[14] S. Shiju George and R. Suji Pramila, “A review of different techniques in cloud computing,” Mater Today Proc, vol. 46, pp. 8002–8008, Jan. 2021, doi: 10.1016/J.MATPR.2021.02.748.

[15] M. Pooyandeh and I. Sohn, “Edge Network Optimization Based on AI Techniques: A Survey,” Electronics 2021, Vol. 10, Page 2830, vol. 10, no. 22, p. 2830, Nov. 2021, doi: 10.3390/ELECTRONICS10222830.

[16] A. Avan, A. Azim, and Q. H. Mahmoud, “A State-of-the-Art Review of Task Scheduling for Edge Computing: A Delay-Sensitive Application Perspective,” Electronics 2023, Vol. 12, Page 2599, vol. 12, no. 12, p. 2599, Jun. 2023, doi: 10.3390/ELECTRONICS12122599.

[17] J. R. Imbien, “Optimizing Collaborative Edge-Cloud Computing: A

Task Allocation Strategy Using Hybrid Genetic Algorithm and Tabu Search Spring 2025".

[18] K. E. Adetunji, I. W. Hofsajer, A. M. Abu-Mahfouz, and L. Cheng, "A Review of Metaheuristic Techniques for Optimal Integration of Electrical Units in Distribution Networks," IEEE Access, vol. 9, pp. 5046–5068, 2021, doi: 10.1109/ACCESS.2020.3048438,.

[19] J. Piri, P. Mohapatra, R. Dey, B. Acharya, V. C. Gerogiannis, and A. Kanavos, "Literature Review on Hybrid Evolutionary Approaches for Feature Selection," Algorithms 2023, Vol. 16, Page 167, vol. 16, no. 3, p. 167, Mar. 2023, doi: 10.3390/A16030167.

[20] Y. Meraihi, A. B. Gabis, A. Ramdane-Cherif, and D. Acheli, "A comprehensive survey of Crow Search Algorithm and its applications," Artif Intell Rev, vol. 54, no. 4, pp. 2669–2716, Apr. 2021, doi: 10.1007/S10462-020-09911-9.

[21] D. Alsadie, "Advancements in heuristic task scheduling for IoT applications in fog-cloud computing: challenges and prospects," PeerJ Comput Sci, vol. 10, p. e2128, 2024, doi: 10.7717/PEERJ-CS.2128.

[22] N. Devi et al., "A systematic literature review for load balancing and task scheduling techniques in cloud computing," Artif Intell Rev, vol. 57, no. 10, pp. 1–63, Oct. 2024, doi: 10.1007/S10462-024-10925-W/METRICS.

[23] A. K. Singh, S. Singh, V. Dubey, P. Kumari, B. Hazela, and V. Singh, "Optimized Task Processing in Cloud Computing Based Frameworks," pp. 843–849, Jun. 2025, doi: 10.1109/INCIP64058.2025.11019301.

[24] S. A. Murad et al., "SG-PBFS: Shortest Gap-Priority Based Fair Scheduling technique for job scheduling in cloud environment," Future Generation Computer Systems, vol. 150, pp. 232–242, Jan. 2024, doi: 10.1016/J.FUTURE.2023.09.005.

[25] W. Xia and L. Shen, "Joint Resource Allocation at Edge Cloud Based on Ant Colony Optimization and Genetic Algorithm," Wirel Pers Commun, vol. 117, no. 2, pp. 355–386, Mar. 2021, doi: 10.1007/S11277-020-07873-3/METRICS.

[26] H. K. Apat, B. Sahoo, V. Goswami, and R. K. Barik, "A hybrid meta-heuristic algorithm for multi-objective IoT service placement in fog computing environments," Decision Analytics Journal, vol. 10, p. 100379, Mar. 2024, doi: 10.1016/J.DAJOUR.2023.100379.

[27] H. Materwala, L. Ismail, and H. S. Hassanein, "QoS-SLA-aware adaptive genetic algorithm for multi-request offloading in integrated edge-cloud computing in Internet of vehicles," Vehicular Communications, vol. 43, p. 100654, Oct. 2023, doi: 10.1016/J.VEHCOM.2023.100654.

[28] M. I. Khaleel, M. Safran, S. Alfarhood, and D. Gupta, "Combinatorial metaheuristic methods to optimize the scheduling of scientific workflows in green DVFS-enabled edge-cloud computing," Alexandria Engineering Journal, vol. 86, pp. 458–470, Jan. 2024, doi: 10.1016/J.AEJ.2023.11.074.

[29] A. Mahjoubi, A. Ramaswamy, and K. J. Grinnemo, "An Online Simulated Annealing-Based Task Offloading Strategy for a Mobile Edge Architecture," IEEE Access, vol. 12, pp. 70707–70718, 2024, doi: 10.1109/ACCESS.2024.3402611.

[30] N. Bacanin, M. Zivkovic, T. Bezdan, K. Venkatachalam, and M. Abouhawwash, "Modified firefly algorithm for workflow scheduling in cloud-edge environment," Neural Comput Appl, vol. 34, no. 11, pp. 9043–9068, Jun. 2022, doi: 10.1007/S00521-022-06925-Y/FIGURES/9.

[31] X. Hua, Y. Jingling, and W. Nana, "A Budget-Constrained Energy-Efficient Scheduling Algorithm on Cloud-Edge Collaborative Workflows," 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2022, pp. 432–437, 2022, doi: 10.1109/CSCWD54268.2022.9776086.

[32] A. Shankar and A. Kuamari, "QoS-aware Scheduling of Periodic Real-time Task Graphs on Heterogeneous Pre-occupied MECs," Jun. 2025, Accessed: Jun. 21, 2025. [Online]. Available: https://www.arxiv.org/pdf/2506.12415

[33] D. Sahu et al., "Beyond boundaries a hybrid cellular potts and particle swarm optimization model for energy and latency optimization in edge computing," Sci Rep, vol. 15, no. 1, pp. 1–22, Dec. 2025, doi: 10.1038/S41598-025-90348-X;SUBJMETA=117,258,639,705;KWRD=COMPUTER+SCIENCE,INFORMATION+TECHNOLOGY.

[34] T. Hai et al., "Task scheduling in cloud environment: optimization, security prioritization and processor selection schemes," Journal of Cloud Computing, vol. 12, no. 1, pp. 1–12, Dec. 2023, doi: 10.1186/S13677-022-00374-7/TABLES/3.

[35] Y. T. Chuang and Y. T. Hung, "A real-time and ACO-based offloading algorithm in edge computing," J Parallel Distrib Comput, vol. 179, p. 104703, Sep. 2023, doi: 10.1016/J.JPDC.2023.04.004.

[36] H. Topcuoglu, S. Hariri, and M. Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 3, pp. 260–274, Mar. 2002, doi: 10.1109/71.993206.

[37] R. Skinderowicz, "Improving Ant Colony Optimization efficiency for solving large TSP instances," Appl Soft Comput, vol. 120, p. 108653, May 2022, doi: 10.1016/J.ASOC.2022.108653.

[38] R. Sh. Othman and Ibrahim Mahmood Ibrahim, "A review of exploring recent advances in ant colony optimization: applications and improvements," International Journal of Scientific World, vol. 11, no. 1, pp. 114–122, Mar. 2025, doi: 10.14419/S0SJGQ84.

[39] S. Lipsa and R. Dash, "SLA-based task scheduling in cloud computing using randomized PSO algorithm," Jul. 2024.

[40] C. P. Shabariram and P. P. Ponnuswamy, "Task offloading in edge computing using integrated particle swarm optimization and genetic algorithm," Advances in Science and Technology Research Journal, vol. 19, no. 1, pp. 371–380, 2025, doi: 10.12913/22998624/195658.

[41] F. S. Prity, M. H. Gazi, and K. M. A. Uddin, "A review of task scheduling in cloud computing based on nature-inspired optimization algorithm," Cluster Comput, vol. 26, no. 5, pp. 3037–3067, Oct. 2023, doi: 10.1007/S10586-023-04090-Y.

[42] K. Sathupadi, "COMPARATIVE ANALYSIS OF HEURISTIC AND AI-BASED TASK SCHEDULING ALGORITHMS IN FOG COMPUTING: EVALUATING LATENCY, ENERGY EFFICIENCY, AND SCALABILITY IN DYNAMIC, HETEROGENEOUS ENVIRONMENTS," Quarterly Journal of Emerging Technologies and Innovations, vol. 5, no. 1, pp. 23–40, Jan. 2020, Accessed: Jun. 21, 2025. [Online]. Available: https://vectoral.org/index.php/QJETI/article/view/133

[43] D. Alsadie, "Advancements in heuristic task scheduling for IoT applications in fog-cloud computing: challenges and prospects," PeerJ Comput Sci, vol. 10, 2024, doi: 10.7717/PEERJ-CS.2128.

[44] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges," IEEE Communications Surveys and Tutorials, vol. 20, no. 1, pp. 416–464, Jan. 2018, doi: 10.1109/COMST.2017.2771153.

[45] C. Mouradian, S. Kianpisheh, M. Abu-Lebdeh, F. Ebrahimnezhad, N. T. Jahromi, and R. H. Glitho, "Application Component Placement in NFV-Based Hybrid Cloud/Fog Systems with Mobile Fog Nodes," IEEE Journal on Selected Areas in Communications, vol. 37, no. 5, pp. 1130–1143, May 2019, doi: 10.1109/JSAC.2019.2906790.

[46] N. M. Dankolo et al., "Optimizing resource allocation for IoT applications in the edge cloud continuum using hybrid metaheuristic algorithms," Sci Rep, vol. 15, no. 1, pp. 1–23, Dec. 2025, doi: 10.1038/S41598-025-97648-2;SUBJMETA=1042,117,258,639,705;KWRD=COMPUTATIONAL+SCIENCE,COMPUTER+SCIENCE,INFORMATION+TECHNOLOGY.

[47] G. Garai and B. B. Chaudhurii, "A novel hybrid genetic algorithm with Tabu search for optimizing multi-dimensional functions and point pattern recognition," Inf Sci (N Y), vol. 221, pp. 28–48, Feb. 2013, doi: 10.1016/J.INS.2012.09.012.

[48] Y. Zheng, Y. Chen, C. Tan, Y. Yang, C. Shu, and L. Chen, "Optimization model for vehicular network data queries in edge environments," Journal of Cloud Computing, vol. 13, no. 1, pp. 1–14, Dec. 2024, doi: 10.1186/S13677-024-00705-W/TABLES/2.

[49] Z. Nan, L. Wenjing, L. Zhu, L. Zhi, L. Yumin, and N. Nahar, "A new task scheduling scheme based on genetic algorithm for edge computing," Computers, Materials and Continua, vol. 71, no. 1, pp. 843–854, 2022, doi: 10.32604/CMC.2022.017504.

[50] S. Kaur and A. Verma, "An Efficient Approach to Genetic Algorithm for Task Scheduling in Cloud Computing Environment," International

Journal of Information Technology and Computer Science, vol. 4, no. 10, pp. 74–79, Sep. 2012, doi: 10.5815/IJITCS.2012.10.09.

[51] F. Xu, Z. Qin, L. Ning, and Z. Zhang, "Research on computing offloading strategy based on Genetic Ant Colony fusion algorithm," Simul Model Pract Theory, vol. 118, p. 102523, Jul. 2022, doi: 10.1016/J.SIMPAT.2022.102523.

[52] H. Nouhas, A. Belangour, and M. Nassar, "A Weighted Scoring Model Analysis of Cloud Storage Services: Comparing AWS, Azure, and Google Cloud Platform," International Journal of Engineering Trends and Technology, vol. 73, no. 6, pp. 350–370, Jun. 2025, doi: 10.14445/22315381/IJETT-V73I6P129.

[53] S. U. Mushtaq, S. Sheikh, and S. M. Idrees, "Enhanced priority based task scheduling with integrated fault tolerance in distributed systems," International Journal of Cognitive Computing in Engineering, vol. 6, pp. 152–169, Dec. 2025, doi: 10.1016/J.IJCCE.2024.12.006.