future internet

MDPI

Review

An Analysis of Methods and Metrics for Task Scheduling in Fog Computing

Javid Misirli * and Emiliano Casalicchio *

Computer Science Department, Sapienza University of Rome, 00185 Rome, Italy
* Correspondence: misirli@di.uniroma1.it (J.M.); casalicchio@di.uniroma1.it (E.C.)

Abstract: The Internet of Things (IoT) uptake brought a paradigm shift in application deployment. Indeed, IoT applications are not centralized in cloud data centers, but the computation and storage are moved close to the consumers, creating a computing continuum between the edge of the network and the cloud. This paradigm shift is called fog computing, a concept introduced by Cisco in 2012. Scheduling applications in this decentralized, heterogeneous, and resource-constrained environment is challenging. The task scheduling problem in fog computing has been widely explored and addressed using many approaches, from traditional operational research to heuristics and machine learning. This paper aims to analyze the literature on task scheduling in fog computing published in the last five years to classify the criteria used for decision-making and the technique used to solve the task scheduling problem. We propose a taxonomy of task scheduling algorithms, and we identify the research gaps and challenges.

Keywords: fog computing; task scheduling; internet of things; machine learning; deep learning; cloud computing

Citation: Misirli, J.; Casalicchio, E. An Analysis of Methods and Metrics for Task Scheduling in Fog Computing. *Future Internet* **2024**, *16*, 16. https://doi.org/10.3390/fi16010016

Academic Editor: Antonio Esposito

Received: 31 October 2023
Revised: 17 December 2023
Accepted: 20 December 2023
Published: 30 December 2023

1. Introduction

The current era has witnessed a paradigm shift in the landscape of the information and communication technology industry with the emergence of the Internet of Things (IoT). IoT represents a groundbreaking technological revolution that has extended Internet connectivity beyond the boundaries of traditional smart devices such as smartphones and tablets, enabling a wide range of appliances, including sensors, machines, and vehicles, to connect and communicate seamlessly. This revolutionary phenomenon has paved the way for various applications and services, including healthcare, medical treatment, traffic control, energy management, vehicle networks, and many others [1].

With its centralized processing capabilities, cloud computing faces significant challenges in meeting the performance requirements of Internet of Things (IoT) applications [2]. This is primarily due to the limited bandwidth and high latency between the cloud and IoT devices. However, a promising solution, known as fog computing, was introduced by Cisco in 2012, establishing a computing continuum between remote cloud datacenters and IoT devices. Fog computing extends the cloud to the network's edge, providing a platform for executing latency-sensitive tasks in fog servers close to the edge devices. Delay-tolerant or compute-intensive IoT applications can be placed to the cloud.

In this context, selecting the appropriate fog node to offload the computation is paramount. Scheduling decisions (i.e., selecting a fog node for computation offloading) consider many factors, like energy consumption, limited resource availability, and low response time. Task scheduling in fog computing has been widely investigated in recent years [3–5]; the problem has been solved using different techniques, and various optimization criteria have been adopted. This study aims to sort out the state-of-the-art literature to answer the following research questions:

RQ1: What performance-related metrics drive the task scheduling decisions in fog computing?

RQ2: What techniques are used to solve the task scheduling problem in fog computing?
RQ3: What are the open challenges of task scheduling in fog computing?

Unlike previous surveys, our review takes a contemporary, comprehensive approach, addressing the latest developments and research directions in fog task scheduling. We reviewed many recent publications and learned about emerging technologies and real-world uses from various sources. A key innovation introduced through our literature review is the development of a novel taxonomy and classification framework. This framework integrates state-of-the-art technologies and classifies the current task-scheduling techniques. Researchers can better grasp the changing fog computing landscape with its help, as it offers a comprehensive reference guide and acts as a dynamic roadmap.

In this research paper, we conduct a comprehensive literature review of the existing research work done in the fog computing task scheduling field. Key contributions are:

- The definition of a taxonomy for task scheduling algorithms in fog computing, which classifies them along two dimensions: the technique for solving the scheduling problem and the metric used to make scheduling decisions.
- A comprehensive literature review of existing scheduling algorithms, particularly focusing on intelligent dynamic scheduling techniques based on machine learning, fuzzy logic, reinforcement learning, and deep reinforcement learning, describing their strengths and weaknesses.
- Identification of the research gaps and challenges for task scheduling in fog computing for future research work in this field.

The remainder of this paper is organized as follows. Section 2 explains the concept of fog computing, its architecture, and its characteristics. The task scheduling problem is fog computing is formulated in Section 3. Section 4 introduces the research methodology adopted. The categorization of the literature is proposed in Section 5. The analysis of the optimization criteria is presented in Section 6, and the analysis of problem solution techniques is discussed in Section 7. Research gaps and challenges are presented in Section 8. Finally, Section 9 concludes the paper.

2. Fog Computing: Architecture and Characteristics

The fog computing architecture consists of three layers [6]: the Edge layer, the Fog layer, and the cloud layer (cf. Figure 1).

At the Edge layer are the edge nodes, devices such as mobile phones, sensors, smart vehicles, smart watches, etc. Edge nodes sense and capture data in diverse geographical locations, eventually pre-process data, and finally send data to the Fog layer for processing or storage.

In the Fog Layer are located the Fog nodes, which can be routers, access points, switches, gateways, firewalls, or dedicated servers [7]. Fog nodes provide computation and storage services and are organized into fog colonies. A Fog Colony is managed by a control node, which is in charge of distributing the colony's workload among the Fog nodes, checking the Fog nodes' health state, and exchanging colony status information with other controllers to eventually use computing and storage services in remote colonies, e.g., when the colony capacity is exhausted or for moving edge nodes. A Fog controller can also use cloud computing and storage services. A Fog colony serves a set of edge nodes that can change over time.

The cloud layer offers scalable computational services and highly available, durable, and large-scale storage solutions. A fog controller uses the cloud layer for storing data that is not needed close to users or performing intensive computations not requiring low latency [8].

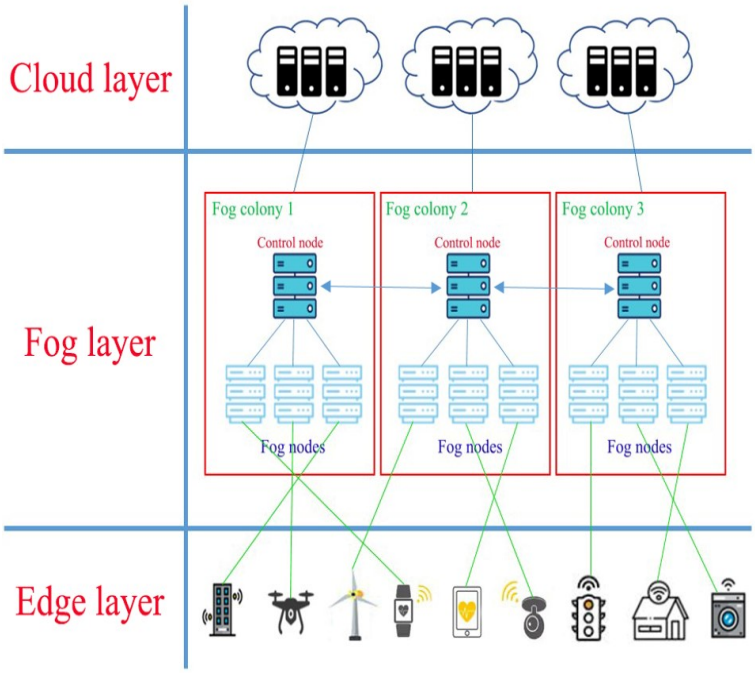

Figure 1. Fog computing architecture.

In this paper, we assume the fog nodes execute IoT applications providing services to the edge nodes. We consider an IoT application composed of multiple software components (tasks) that can be executed at the edge, in the fog, or on the cloud layers. When deploying an IoT application to cater to a single edge node or a group of edge nodes, the control node of the closest Fog colony takes scheduling decisions for the application's tasks. Scheduling decisions are based on the computational demand and available resources, and on functional and non-functional requirements.

The architecture described above generalizes what has been proposed in the literature. According to [3], the fog computing architecture is a heterogeneous environment containing multiple fog layers and the cloud; fog and cloud servers are collaborative, and tasks are dependent.

The authors of [9] depict the architecture integrated with mobile edge computing (MEC). The model consists of user level, edge level, and remote level. At the edge level, devices are multiple, and requested tasks are heterogeneous. MEC contains virtualization technology that provides computing, storage, and network resources. The edge layer comprises a homogeneous environment and a single layer. According to the model, an application has dependent tasks. One main drawback is that it is not robust in case of broken network connections or insufficient power.

A three-layer network architecture consisting of an IoT device layer, a fog layer, and a cloud layer has also been proposed in [10]. The IoT layer routes requests from IoT devices to the fog layer using intelligent gateways. The fog layer consists of a collection of fog nodes with a fog controller or broker. The fog controller distributes tasks between the fog and cloud layers. The fog controller also manages the scheduling of tasks across the fog nodes and coordinates their available resources. The cloud layer consists of data centers with significant computing power. The tasks are characterized by the fact that they are indivisible, non-preemptible, and independent, which makes them unsuitable for being divided into sub-tasks. Fog nodes are heterogeneous, meaning each node differs from others.

Characteristics

The main fog computing characteristics are geographical distribution [11], decentralization [1], heterogeneity [12], real-time interaction, low latency [13], and mobility [14].

Fog computing is a decentralized computing infrastructure in which data, computing, storage, and applications are geographically distributed and located between the data source and the cloud [11]. There is no central server to manage computing resources and services. Therefore, fog nodes are self-organizing and collaborate to provide end users with real-time IoT applications [1].

Fog nodes are deployed and operated independently, forming a heterogeneous computation and storage network. Fog computing exhibits versatility in operating across multiple platforms [12].

Fog computing sustains real-time interaction rather than batch processing. Real-time refers to a system required to respond within a specific time frame. This category of applications includes real-time e-health, traffic data transmission, aviation operations, and critical industrial process monitoring systems [13]. Fog computing distinguishes itself from cloud-based paradigms by offering efficient and rapid real-time responses.

Mobility is an intrinsic trait of several fog applications to improve users' experience [14]. Many IoT devices are moving objects (like vehicles), demanding connected service through their path. Hence, decision engines should oversee the process of migrating application components and select appropriate replacement servers as necessary. Due to the mobility of fog nodes, the selection of tasks for fog nodes has been made dynamically.

Low latency is critical for delay-sensitive applications such as video surveillance, live streaming, and data analytics. Fog computing enables the emergence of the latency-sensitive Internet of Things network to support real-time applications. The fog is closer to end devices. Therefore, the latency in response is much less than that of cloud computing [13].

3. Task Scheduling Problem in Fog Computing

Task scheduling is a problem addressed in many papers, e.g., [15–21]. In [15], the authors formulated task scheduling problems as Markov Decision Processes (MDP). Their novel solution helps to place the task in the fog nodes or the cloud data center. The objective of the placement problem addressed is the minimization of time and cost of delivery of services in fog-based IoT applications. Long Mai et al. [17] considered the task scheduling problem as selecting appropriate fog nodes to minimize long-term computation latency. Due to the high complexity of the real-time task, deploying all tasks in the fog environment is challenging. The authors introduce a method founded on reinforcement learning, especially using an evolutionary strategy for task distribution between fog servers. The research defines the training environment as the system consisting of fog servers and their respective buffers. Researchers use the buffer utilization of available fog nodes to represent the environment states. The authors of [20] stress the importance of dynamic task scheduling, which becomes crucial to utilize the resources at fog and cloud layers to improve QoS. In their approach, after making an allocation choice, the fog server will be responsible for providing the resources for a particular task.

There are studies where the task scheduling problem is solved in specific application domains, like smart manufacturing, vehicular applications, and mobile edge computing. Yin et al. [21] present a new task-scheduling model computing that can be applied in smart manufacturing, where tasks can support essential network functions or new services and applications. In such a delay-sensitive and resource-constrained domain, task scheduling became a challenge. Wang et al. [18] formulated the task scheduling problems for fog computing applied in the smart production domain and proposed a hybrid heuristic-based algorithm. They described the problem as independent tasks assigned to fog nodes, in which task scheduling aims to minimize the time and energy consumption costs through a reasonable allocation. They aim to allocate multiple tasks to multiple fog nodes based on the given constraints. Karimi et al. [19] devised deep reinforcement learning-based solutions for vehicular application tasks. The task placement is chosen between Multi-access Edge Computing (MEC) and the cloud. The main challenge is to find an optimal task scheduling strategy and maximize the vehicular application task's acceptance rate.

As a result, they got less response time for an application. Zheng et al. [16] present a comprehensive investigation of the problem of task migration in mobile edge computing systems. The fundamental issue is the increase in mobility possibilities and the dynamic changes in mobility patterns in connection with autonomous vehicles. A DQN-based task scheduling algorithm is proposed, in which an agent can learn an optimal task migration policy from past experience, even without knowing the users' mobility patterns.

This work considers scheduling IoT applications in a fog computing environment. An IoT application, represented in Figure 2, provides data, information, or actuating functions to the requesting clients. The application is an ensemble of software components that perform dependent or independent tasks [22]. The objective of the task scheduling problem is to assign tasks to computational nodes to meet the IoT application's QoS requirements and consider the fog nodes' future processing commitments [23]. In the fog environment model, we consider computational nodes to be constrained regarding CPU, memory, disk, bandwidth, or battery lifetime. Figure 3 shows the scheduler can assign tasks to edge, fog, or cloud nodes. Although tasks can be classified as dependent or independent, we consider independent tasks with heterogeneous resource demand.

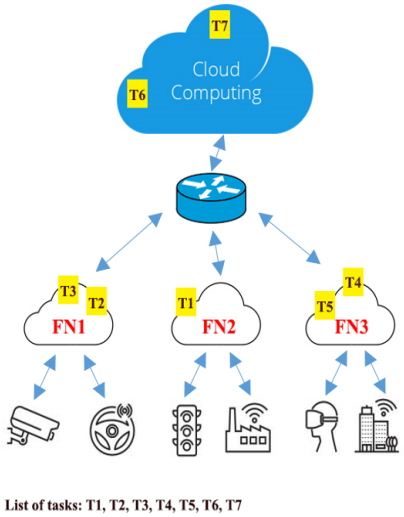

Figure 2. Placement of application tasks.

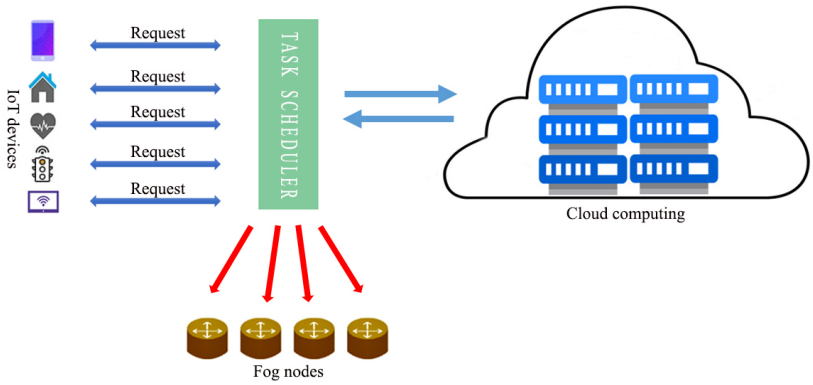

Figure 3. Task scheduling architecture in fog environment.

Objectives and Constraints

IoT applications run in constrained environments and, at the same time, have performance and availability requirements. When it comes to the scheduling problem, the scheduler has to achieve its own objectives (e.g., minimization of the timespan or maximization of the resource usage) considering both constraints of the environment (e.g., CPU and memory capacity) and the application's requirements (e.g., response time and availability), which

translate into the problem's constraints. Performance metrics measure schedulers' objectives and applications' requirements. In Section 5, we will analyze the literature regarding performance metrics optimized by the scheduler. Table 1 shows what the applications' and environments' constraints are used in the literature.

Table 1. Task scheduling problem's constraints.

Application's Constraints	Use in Literature
Response time	[3,11,19,20,24–27]
Completion time	[3,4,28–31]
Execution time	[21,31–38]
Processing time	[30]
Delay	[5,8,10,11,15–18,20,21,27,28,32,37,39–54]
Availability	[10,30,43]
Environment's constraints	
Energy consumption	[3,4,7,10,15,18,20,24,26,28,29,36,39,41,42,44,46, 47,49–51,55–58]
The amount of CPU available	[15,17,30,32,33,35,36,39,43,46,49,55,56]
The amount of memory available	[15,32,33,35,47,51,56]
The amount of storage	[15,20,28,30,32,33,36,40,47]
Battery lifetime	[28,49,57]
Computing capacity of fog node	[28]
Data size of task	[43]
Network usage	[44,46]

4. Research Methodology

As mentioned in the introduction, this paper aims to answer the following research questions: What performance-related metrics drive the task scheduling decisions in fog computing? What techniques are used to solve the task scheduling problem in fog computing? and What are the open challenges of task scheduling in fog computing?

In undertaking our research, we employed a Systematic Mapping Study (SMS) methodology with the primary objective of systematically organizing and categorizing literature spanning the last five years in fog computing [59,60]. Our literature search commenced by identifying seminal works in the field, serving as the foundational starting point for our study. From this initial stage, we systematically traced forward citations using the snowballing technique, a key component of the SMS methodology [61]. At each stage of the mapping process, we meticulously considered various dimensions, encompassing decision-making criteria and techniques utilized in addressing the task scheduling problem within fog computing. The SMS methodology facilitated the systematic categorization and visualization of the literature, offering a comprehensive overview of the research landscape [59]. The resulting taxonomy of task scheduling algorithms and the identification of research gaps and challenges stand as direct outcomes of our SMS methodology. This approach ensured a rigorous and structured analysis of the literature, providing valuable insights into the evolving domain of fog computing task scheduling.

The search string used to gather papers is ("task scheduling" OR "job scheduling") AND ("fog computing" OR "edge computing") AND ("IoT" OR "Internet of Things") AND ("challenges" OR "issues" OR "solutions").

We used the Boolean operators "OR" and "AND" to refine the results we obtained. The use of "AND" limited the search results to articles containing both keywords, while "OR" expanded the search scope to include articles containing either.

Our literature search, conducted from 2018 to October 2023, employed a targeted search string, applied to major Digital Libraries (DLs). The initial search returned a substantial pool of papers totaling 200. To ensure a robust dataset, we implemented a systematic deduplication process. Following this meticulous process, we arrived at a final set of 71 papers for detailed analysis. The temporal distribution of these articles is presented in

Figure 4. Notably, about 50% of the papers were published in 2021–2022, reflecting the contemporary nature of our selected studies.

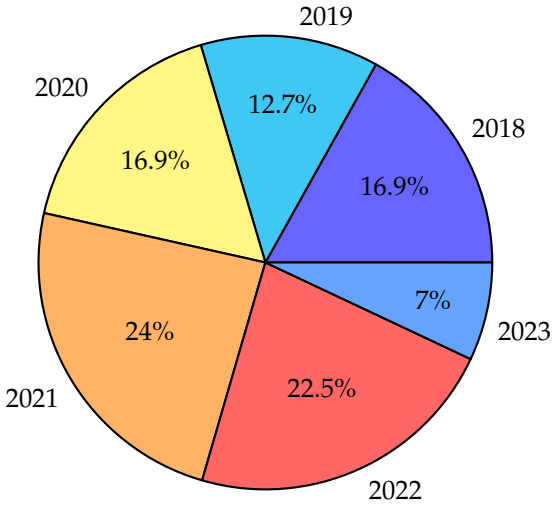

Figure 4. Year of publication of the selected papers.

Our screening process is structured into two distinctive stages to ensure a meticulous and unbiased selection of studies [62]:

1. Initial Screening: Deduplication: rigorous deduplication is conducted to eliminate duplicate records from the database, ensuring a clean dataset. Title and abstract examination: The initial screening phase involves a detailed examination of titles and abstracts. This preliminary assessment is essential to minimize biases, including potential publication bias.
2. Secondary Screening: Full study screening and critical appraisal: The second phase comprises a comprehensive evaluation of the full texts of selected studies. We meticulously followed predefined inclusion and exclusion criteria throughout the screening process, as detailed in Table 2. These criteria served as a robust framework for assessing the relevance and quality of each identified study.

Table 2. Selection criteria.

Inclusion Criteria	Exclusion Criteria
Written in English	In Press Articles
Areas: Computer Science and Engineering	Not written in English
Type: Conference Proceedings, Journal Article	Short papers, Extended abstract
	Papers published before 2018

Figure 5 highlights that a significant portion of our final set of papers has been published in IEEE journals, indicating both the credibility and relevance of our selected studies.

We employed a two-fold approach to comprehensively survey the literature on task scheduling in fog computing, integrating the snowballing technique and the Quasi-Gold Standard [63]. This method allowed us to explore both direct and indirect citation relationships, contributing to a nuanced perspective on the task scheduling landscape. The snowballing technique was applied from August 2021 to January 2022. Starting with seminal works, we systematically traced forward citations over three iterations. Each round involved meticulous examination of references in identified studies. By January 2022, we concluded the process, ensuring a comprehensive and up-to-date exploration of task scheduling in fog computing.

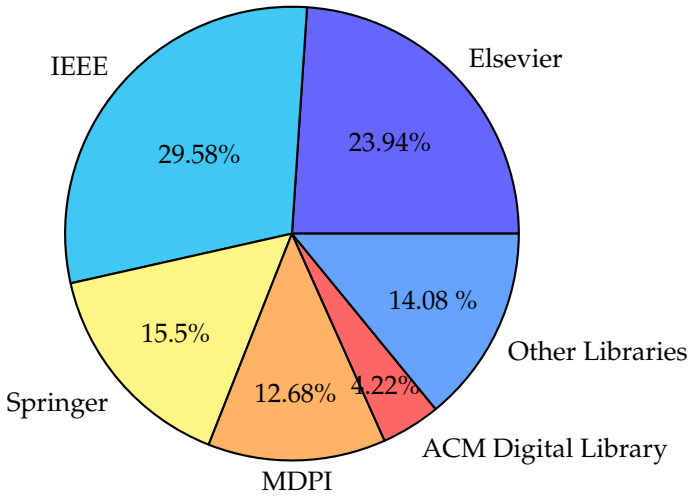

Figure 5. Publisher of the selected papers.

5. Taxonomy for Task Scheduling Algorithms in Fog Computing

To design task scheduling algorithms, selecting appropriate metrics for optimization criteria and choosing the appropriate problem formulation and solution techniques is important. Therefore, we have decided to classify the literature along two dimensions: metrics for optimization criteria and problem-solution techniques.

5.1. Optimization Criteria

The purpose of performance evaluation metrics in fog computing is to gauge the effectiveness and efficiency of the system and ensure that it meets the expected quality of service for end users [2]. In addition, they can potentially define the utility or function that the fog provider wants to optimize. Figure 6 shows the metrics considered.

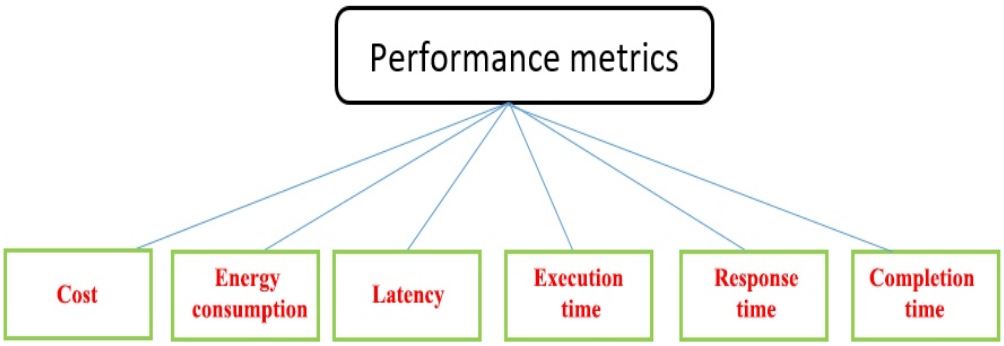

Figure 6. Category of performance metrics used in the optimization criteria.

5.1.1. Cost

Costs can be related to computation, communication, resource usage (other than CPU and network), and energy consumption.

Computation cost. Performing a task on a fog node or device is called computation cost. It covers the time, energy, and resources used to complete the computation [15]. Depending on the difficulty of the task and the fog node's capabilities, computation costs might change.

In the Cloud–Fog system, carrying out a task results in costs tied to processing, memory consumption, and bandwidth utilization. The cost estimated when node N_i processes the task T_k is expressed as [33]:

$$Cost(T_i^k) = c_p(T_i^k) + c_m(T_i^k) + c_b(T_i^k)$$

Processing cost is defined as:

$$c_p(T_k^i) = c_1 * Exect(T_k^i)$$

where c_1 is the CPU usage cost per time unit in node N_i, and Exect(T_k^i) is the execution time of T_k processed in node N_i. Letting c_2 be the memory usage cost per data unit in node N_i and Mem(T_k) be the memory required by task T_k, the memory usage cost is:

$$c_m(T_k^i) = c_2 * Mem(T_k^i)$$

Task T_k processed in node N_i needs an amount of bandwidth $Bw(T_k)$, which is the sum of input and output file size. Let c_3 be the bandwidth usage cost per data unit; the bandwidth usage cost is defined as follows:

$$c_b(T_k^i) = c_3 * Bw(T_k^i)$$

Total cost for all tasks to be executed in Cloud–Fog system is calculated as follows:

$$TotalCost = \sum_{T_k^i \in NodeTasks} Cost(T_k^i)$$

Communication cost. Tasks may need to be transferred between fog nodes for processing in a distributed system like IoT and fog computing [64]. Communication cost includes the overhead of transferring data, such as bandwidth usage, latency, and energy consumption between nodes [40].

Resource cost. Fog nodes have constrained CPU, memory, and storage capabilities. In the context of fog computing, resource cost entails efficiently assigning and utilizing computing resources essential for the effective execution of tasks. This encompasses the utilization and consumption of a range of resources, such as computational power, storage, and communication bandwidth. The term reflects the comprehensive expenditure needed to ensure efficiency within a fog computing environment.

Energy cost. Energy efficiency is a crucial factor for many IoT and fog computing devices as they are battery-powered. The cost of energy consumption is dependent on the amount used for communication and processing [56].

5.1.2. Energy Consumption

Different models for energy consumption have been proposed in the literature. The more relevant ones are reported here.

Energy consumption derives from different activities executed by a fog node, such as computation and data transmission. For example, if the j-th edge device wants to transfer the i-th task at time t, the total energy consumption is calculated as the sum of the energy required for computation (E_{comp}) and the energy needed for data transmission (E_{tran}), defined as follows [49]:

$$E_{\text{comp}}(i,j,t) = \alpha_1 c_i (f_j)^2$$

$$E_{\text{tran}}(i,j,t) = \alpha_2 s_i / r(t)$$

$$E_{\text{total}} = E_{\text{comp}} + E_{\text{tran}}$$

where α_1 and α_2 are the coefficient of energy consumption in computing and transmission, f_j is the CPU clock frequency of mobile devices, $r(t)$ is the network transmission rate at time t, and s_i and c_i are the content size and the amount of computation.

In [41], the energy consumed to process as task i in a fog node is denoted as E_f. Moreover, as the task is transmitted from the end-user to the fog node, the energy consumption for data transmission and reception is denoted by E_t^i and E_r^i, respectively. The authors express the total cost at the fog node side as follows:

$$E_f = E_t^i + E_r^i + \alpha C_f^i$$

where α C_f^i is the weighted transmission energy with α being the relative weight and C_f^i being the relative cost for processing task i at the fog node.

5.1.3. Latency

Latency in fog computing task scheduling signifies the time delays during different stages of task execution. It is a vital performance metric influencing overall system responsiveness and efficiency. Reducing latency is essential for meeting real-time requirements, enhancing user satisfaction, and optimizing resource use in fog computing. Scheduling algorithms strive to minimize latency by efficiently assigning tasks, handling communication, and adapting to dynamic system changes [46].

5.1.4. Response, Execution, and Completion Time

Response time is the time between sending a request to the server and completing the task execution. An increase in the response time will cause latency and delays. Response time includes both the network latency and the time for the computation. The response time of processing task t_i on fog node f_j is calculated as the sum of the specified node's execution time plus the task's transmission time from the source to the destination [46]:

$$RT(X_{ij}) = Et(X_{ij}) + Tr t_{total}(X_{ij})$$

where X_{ij} refers to the assignment of task t_i to fog node f_j, $Tr t_{total}(X_{ij})$ is the total transmission time of tasks from the edge device to the appropriate fog node f_j, and $Et(X_{ij})$ is the execution time of the task in the fog node.

If the task is processed locally, the response time equals the local serving time. If a task is scheduled, the total response time is the sum of the task transmission time to the fog node through the wireless network, the processing time on the fog node, and the transmission time back to the IoT devices. If the fog node is overloaded, the processing time at the fog is replaced by the transmission time between the fog node and the remote cloud, the processing time on the cloud and the transmission time back to the fog node [65].

Execution time is the total run time for the fog application, and it is measured in milliseconds. For task computing on a fog server, the processing of task n is divided into two phases. The first phase is the transmitting phase where the industrial sensor sends task data to fog through wireless transmission. The second phase is the fog-computing phase where task n is executed in the fog. Fog processing delay for each task is the sum of the delay due to transmitting task data through wireless links and fog-server computing time. The formula is as follows [4]:

$$T_{n,m}^f = \frac{d_{n,m}}{r_{n,m}} + \frac{C_{n,m}}{f_{n,m}^f}$$

where $f_{n,m}^f$ the computational demand of a fog server for a particular task (n) as perceived from the perspective of a specific fog node (m). Here, n indicates task number, and m indicates the fog node number. $C_{n,m}$ is the completion time, $d_{n,m}$ is the input data size, and $r_{n,m}$ is the data transmission rate for task n.

Completion time of a task n is determined as the time it takes to complete execution. It includes the wait and processing times of the task. The completion time $CT_{n,m}$ of a task n can be expressed as follows [4]:

$$CT_{n,m} = RT_{n,m} + T_{n,m}$$

where $T_{n,m}$ defines the execution time of the task n and $RT_{n,m}$ represents ready time that the earliest completion time for all its dependent tasks.

5.1.5. Availability

This is the ability of a system to ensure the data are available with the desired level of performance in normal as well as in fatal situations, excluding scheduled downtime. It is a ratio of Mean Time Between Failure (MTBF) to reliability (Mean Time Between Failure + Mean Time To Repair). The following formula can be applied to calculate availability [66]:

$$Reliability = MTBF + MTTR$$

$$Availability = MTBF / Reliability$$

5.2. Problem Solution Techniques

In a fog-based IoT environment, application service requests can be made at the level of the IoT device, the fog servers, or the cloud data centers [67]. Proper task scheduling algorithms determine the best allocation of incoming tasks for each application service, which may consist of a sequence of tasks. We propose a task-scheduling algorithm classification based on the technique used for problem formulation and solution. The categorization is presented in Figure 7.

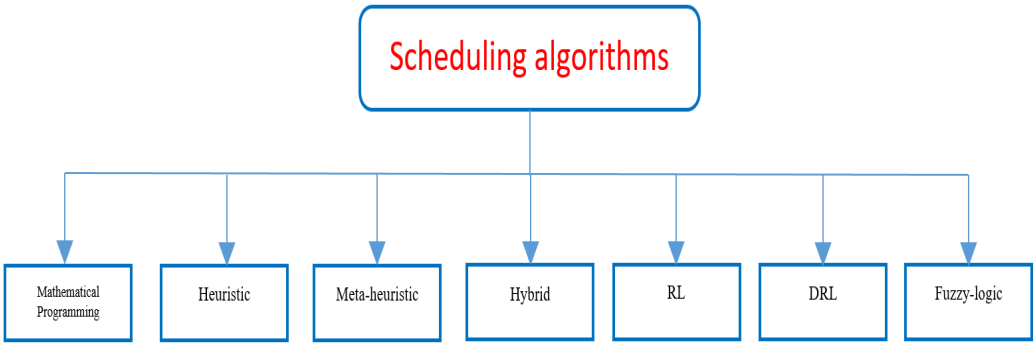

Figure 7. Category of task scheduling algorithms.

5.2.1. Mathematical Programming-Based Approaches

These methods frame the scheduling problem as an optimization challenge.

For task scheduling in fog computing, two mathematical programming models are used: Integer Linear Programming (ILP) and Mixed Integer Linear Programming (MILP). However, these approaches demand significant computational resources. They may be affected by the size and intricacy of the problem, thus being better suited for small and medium-sized fog computing systems than larger-scale ones.

Integer Linear Programming is a well-known mathematical technique involving developing a mathematical model representing the task scheduling problem as a set of linear equations and constraints. In fog computing task scheduling, ILP can allocate tasks to devices to minimize the overall completion time or maximize the utilization of available resources while also considering constraints related to device capabilities and network conditions [21].

Mixed Linear Integer Programming is frequently utilized to address optimization problems involving discrete and continuous variables and non-linear functions in the constraints or objective function. In fog computing task scheduling, MILP can be used to assign tasks to devices to minimize the overall completion time or maximize the utilization of available resources. MILP can also incorporate constraints related to device capabilities and network conditions, ensuring that the assigned tasks can be executed efficiently.

5.2.2. Heuristic-Based Approaches

Heuristic algorithms offer an alternative optimization technique for task scheduling in fog computing [68]. These algorithms leverage a set of rules to direct the search for an optimal solution, eschewing reliance on a mathematical model. They can assign tasks to

devices based on various criteria, such as device availability, processing power, and network latency. Heuristic algorithms are well-suited for handling large-scale systems where the computational complexity of mathematical models, such as ILP and MILP, can become impractical. Heuristic solutions are also combined with mathematical programming modeling to determine the optimal solution and satisfy a predefined set of constraints and objectives [18].

5.2.3. Meta-Heuristic-Based Approaches

Meta-heuristic algorithms improve a candidate solution iteratively using high-level heuristics that guide the search for a good solution, rather than relying on a mathematical model [34]. Compared to heuristic algorithms, meta-heuristics algorithms can escape local optima and explore a larger portion of the search space, leading to better quality solutions. Some widely used meta-heuristic algorithms in fog computing task scheduling include Genetic Algorithms, Particle Swarm Optimization, and Ant Colony Optimization [3].

5.2.4. Hybrid Approaches

Hybrid task scheduling algorithms in fog computing refer to approaches that combine two or more approaches to achieve better performance. These algorithms leverage the strengths of multiple scheduling techniques, such as heuristics, meta-heuristics, or optimization methods. The hybrid nature allows these algorithms to exploit complementary features of the different methods to improve scheduling decisions [69].

5.2.5. Reinforcement Learning-Based Approaches

Reinforcement Learning is a subset of machine learning that aims to train agents to make decisions that maximize rewards within a given environment. Reinforcement Learning-Based Approaches refer to fog computing task scheduling algorithms that employ Reinforcement Learning principles to optimize the system's performance [70]. These approaches can handle complex scheduling scenarios and are adaptable to the changing conditions of the system.

5.2.6. Deep Reinforcement Learning-Based Approaches

Deep Reinforcement Learning (DRL) integrates reinforcement learning principles with deep neural networks, empowering agents to learn and decide in intricate environments. When implemented in fog computing task scheduling, DRL facilitates adaptive and intelligent decision-making, accommodating fog environments' dynamic and diverse characteristics [19]. This strategy can potentially enhance resource utilization efficiency, minimize latency, energy consumption and boost overall system performance.

This approach can help manage the system's variability and unpredictability in fog computing, such as dynamic workload and resource availability. Deep Reinforcement Learning-based approaches can handle complex scheduling scenarios and adapt to changing system conditions.

5.2.7. Fuzzy Logic Approach

Fuzzy logic is a mathematical framework capable of handling uncertain and imprecise information [30]. In task scheduling, this approach can aid in managing the system's environmental unpredictability and variability, such as fluctuating workload and resource availability, typical of fog computing.

6. Literature Mapping

Optimization Criteria

Table 3 maps the studied literature with the optimization criteria described in Section 5.1 and is used to make scheduling decisions. The metrics authors consider more often are energy consumption, latency, and cost. Less frequently, research work use response time, execution time, completion time, and availability.

Table 3. Decision-making features.

References	Cost	Energy	Latency	Exec. Time	Resp. Time	Completion Time	Availability
[15,37,41,49,50]	✓	✓	✓				
[5,9,16,17,45,48,52–54]			✓				
[32,33,71]	✓						
[10,39,42,44,46]		✓	✓				
[40]	✓						
[3]	✓	✓			✓	✓	
[4,18,28]		✓				✓	
[21,34,68]				✓			
[51]		✓		✓			
[19,25,27]					✓		
[35,38,72]	✓			✓			
[36,73]	✓	✓		✓			
[43,74,75]							✓
[55,57,58]		✓					
[20]	✓	✓	✓		✓		
[24,29]	✓	✓			✓		
[56]	✓	✓					
[47]	✓		✓	✓			
[11]			✓		✓		
[30]			✓			✓	
[31]				✓		✓	
[26]		✓			✓		
[64]	✓		✓				
[6]					✓	✓	

Task scheduling requires different decision-making metrics depending on workload, resource availability, deadlines, and specific optimization goals. The choice of appropriate decision-making metrics for task scheduling also hinges on the nature of the computing environment, the type of tasks being scheduled, and the overall system architecture. Moreover, considerations like real-time requirements, task dependencies, and the dynamic nature of workloads play a crucial role in determining the most effective decision metrics for optimal task scheduling.

Three evaluation metrics (cost, energy, latency) are considered in [15], focusing on minimizing average End-to-End (E2E) service delay, managing computation costs within resource and deadline constraints, and enhancing energy efficiency during periods of heavy task loads. The following metrics have been considered in [20,37,41,49,50] as well.

The authors [3,20] have extensively examined decision metrics. Tuli et al. [3] present a methodology to minimize energy consumption, response time, cost, and completion time within Edge–Cloud environments. Bukhari et al. [20] contribute significantly to comprehending essential criteria by accentuating overall latency, energy consumption, response time, and operational cost reduction. The authors of [5,9,16,17,19,21,25,27,34,40,43,45,48,52–55,57,58,68,71,74,75] have explored the only the effect of one performance metric on scheduling decisions.

The significance of incorporating energy consumption and latency metrics into the decision-making process is demonstrated in [10,39,42,44,46]. Zhao et al. [39] utilized energy consumption and delay metrics within homogeneous fog networks. They considered the energy expended during task execution and the power consumed during task scheduling. Additionally, they established that the average traffic delay for each mobile fog node is directly linked to the average number of unexecuted computation tasks. Other authors offer innovative strategies for fog computing task scheduling, utilizing fuzzy reinforcement learning, centralized DDQN algorithms, and efficient schedulers to reduce energy consumption and latency. The emphasis is on enhancing efficiency in dynamic and resource-constrained environments, with a focus on optimizing performance metrics for IoT devices.

Execution time and cost metrics have been investigated together by [32,33,35,36,38,72]. Tsai et al. [32] examine CPU processing power, CPU usage cost, memory usage cost, and bandwidth usage cost as operating cost factors alongside timespan, representing execution time, in their assessment framework. Their approach focuses on minimizing execution time with higher priority unless there is a tradeoff where minimizing total operating costs becomes more crucial than execution time. The specific value of the tradeoff coefficient (λ) is determined based on factors such as budget constraints or the desired response time level. Considering these collective contributions, the papers provide unique perspectives on optimizing task scheduling in Cloud–Fog environments, with a specific focus on processing large-scale Internet of Things (IoT) applications. The introduction of inventive algorithms such as Time-Cost aware Scheduling (TCaS) and the application of a Deep Reinforcement Learning (DRL)-based solution underscores the emphasis on delicately balancing execution time minimization and effective monetary cost management. This collaborative initiative aims to propel efficiency and resource utilization within highly distributed computing platforms, effectively addressing pivotal challenges in the realm of fog computing and IoT applications.

Scheduling algorithms that aim to reduce, or in general to optimize, the energy consumed by a fog-based system are studied in [3,4,10,15,18,20,24,26,28,29,36,37,39,41,42,44,46,49–51,55–58,73]. Scheduling based on latency is proposed in [5,9–11,15–17,20,30,37,39,41,42,44–50,52–54,64].

Computation cost is considered in [15,40,72,73]. This cost in fog computing task scheduling refers to the resources used for task execution which is a critical factor impacting system efficiency and performance. The cost of communication as decision criteria is studied by [33,40,47,64,73], while in [20,32,33,40,47], the authors studied the influence of resource costs, and in [29,36,41,47,49,56], they consider energy costs. The authors focus on optimizing different costs to enhance the overall efficiency and performance of fog computing task scheduling.

The importance of execution time metrics while building an optimal task scheduling solution is considered in [21,31–36,38,47,51,68,72,73]. Response time is used for scheduling decisions in [3,6,11,19,20,25–27,29]. A few papers [3,4,6,18,28,30,31] take into account completion time metric.

The often-neglected metric is availability, which remains critical in fog computing environments and has been discussed by [43,74,75].

7. Literature Review of Problem Solution Techniques

7.1. Integer Linear Programming Algorithms

Tsai et al. [32] studied a policy for optimal task scheduling that minimizes execution time, conserves network bandwidth, and reduces operating costs. As the size of the problem increases, the computational complexity of the proposed approach experiences a rapid rise, which limits its effectiveness (Table 4).

Fonseca et al. [31] proposed also an ILP formulation of the task scheduling problem whose goal is to minimize the execution time and makespan of fog computing applications. The task scheduler determines where the applications should be executed on the fog or cloud infrastructure.

Table 4. Techniques to solve the task scheduling problem.

Task Scheduling Algorithms	Use in Literature
Integer Linear Programming	[31,32]
Heuristic and Meta-heuristic	[26,34,38,52,53,68]
Hybrid	[18,28,58,69]
Reinforcement Learning	[8,27,54,64,72,74]
Deep Reinforcement Learning	[3,5,15,16,19,35,36,41,43]
Fuzzy Logic	[30,51,71,75]

7.2. Heuristic and Meta-Heuristic-Based Solutions

Canali et al. [53] formulate the scheduling problem as a mapping between edge nodes (e.g., sensors) and fog nodes to ensure a high quality of service (QoS) regarding latency and response time. A genetic algorithm is proposed to find the sub-optimal solution. The limitation of the genetic algorithm is that it may not converge to the optimal solution every time. The analysis conducted by the authors did not include considerations for energy consumption and cost concerns.

Aburukba et al. [52] propose an ILP formulation for the scheduling problem to minimize latency in IoT applications operating in a hybrid fog–cloud computing environment. The optimization problem was solved using a genetic algorithm accounting for various request characteristics, including priority, processing time, and location. A drawback of the solution is that it does not facilitate pre-emptive scheduling.

Souri et al. [34] formulate the task scheduling problem as a nondeterministic polynomial problem solved by means of a population-based meta-heuristic algorithm, which minimizes the total execution time of tasks while fulfilling the Quality of Service (QoS) requirements of Cyber–Physical System applications.

Abdulredha et al. [38] address the task-scheduling problem for the deployment of bag-of-tasks applications on the Cloud–Fog environment. The proposed solution is the Evolutionary Task Scheduling algorithm, which uses an evolutionary approach. Experimental results demonstrate that the algorithm balanced completion time and financial cost for a set of tasks in the Cloud–Fog system. However, the paper does not address several critical issues, including resource utilization, load balancing, and energy efficiency.

Vakilian et al. [26] introduced a population-based meta-heuristic, the Cuckoo Algorithm, to minimize response time and reduce energy consumption costs associated with scheduling in fog networks. Their algorithm has drawbacks, such as neglecting to adjust CPU frequencies for nodes and failing to segment traffic scheduling into multiple requests.

Aburukba et al. [68] proposed a meta-heuristic approach that utilizes the genetic optimization algorithm to allocate computation requests from static IoT devices. Their approach's main objective is to minimize missed deadlines and timely fulfillment of requests using a mixed integer programming optimization model. The proposed model considers various delays, including transmission delays, queuing delays at networking devices, and task execution delays at the fog and cloud layers. Their solution was compared with the simulation environment's round-robin and priority scheduling techniques.

7.3. Hybrid Algorithms

Wang et al. [18] proposed an hybrid heuristic framework incorporating Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) algorithms. It is worth noting that the PSO algorithm is recognized for its quick search speed, whereas the ACO algorithm is particularly known for its ability to achieve greater precision. The authors considered factors such as completion time, energy consumption, and reliability.

Khaleel et al. [69] introduces a hybrid multi-criteria decision-making (Hybrid-MCD) model which minimize application response time by integrating iterative critical path search and layer-based priority rationing (LPR). Moreover, their application placement strategy utilizes a Deadline-aware stepwise Reliability Optimization (DARO) algorithm

to reassign tasks along the non-critical path. This process serves to elevate the overall reliability of the application.

Hosseinioun et al. [58] introduced a hybrid Invasive Weed Optimization and Culture (IWO-CA) evolutionary algorithm to addresses energy-aware task scheduling. The authors consider leveraging the Dynamic Voltage and Frequency Scaling (DVFS) technique to regulate the energy consumption of fog nodes by adjusting their voltage and frequency levels based on workload demand.

The study conducted by Defude et al. [28] aimed to decrease task completion time and energy consumption. Their proposed approach utilizes a population-based meta-heuristic technique crafted to choose the most optimal fog nodes. This novel approach, called Opposition-based Chaotic Whale Optimization Algorithm (OppoCWOA), combines Whale optimization algorithm (WOA), Opposition-based learning, and Chaos theory.

7.4. Reinforcement Learning Based Algorithms

Guo et al. [54] propose a reinforcement learning-based task-scheduling scheme for vehicular edge computing networks. The scheme aims to minimize the processing delay of all tasks by assigning the tasks to the edge device, a mobile edge computing servers (fog node), or the cloud server. While their deep Q-learning algorithm may enhance the effectiveness of solutions for complex scenarios with many vehicles, the scheduling approach may not adapt well in dynamic environments.

Peng et al. [27] have employed an alternative approach based on reinforcement learning (RL) to reduce latency and enhance the security for both fog nodes and Internet of Things (IoT) tasks.

Cheng et al. [72] formulate the task scheduling problem in fog computing as a DAG task scheduling problem in a heterogeneous computing system. Reinforcement learning is used to solve the problem. They consider three main performance metrics in the evaluation phase: schedule length ratio, speedup, and scheduling algorithm's running time.

Farhat et al. [74] presented a novel cloud scheduling approach utilizing a reinforcement learning-based model, which effectively employs fog resources to alleviate the burden on the cloud. Their study primarily focuses on understanding user request behavior, and their proposed decision model accounts for temporal and spatial factors. However, their placement decision policy requires refinement concerning the selection of services to incorporate.

Dehury et al. [8] proposed a novel concurrent reinforcement learning placement approach for optimizing service delivery in fog and cloud environments. The proposed model incorporates the users' attributes, such as distance and service usage intensity, as constraints in the placement process. They are motivated by a need to address service delivery issues by achieving optimal distribution of service loads between fog and cloud computing platforms while maintaining a high service quality during service delivery.

Pandit et al. [64] has developed an alternative task scheduling policy that relies on reinforcement learning principles. This study aims to reduce both the overall computational latency and communication costs. The proposed solution relies on a two-level neural network (NN). The first-level neural network assesses whether the data stream can be processed within the resource-constrained edge/fog environment or whether it should be sent directly to the cloud. The second level NN then schedules the tasks received from the first level NN to the fog layer, considering the available fog devices.

7.5. DRL-Based Scheduling Algorithms

Zheng et al. [16] present a solution to the task migration problem using deep reinforcement learning. The problem involves determining the optimal time and location to move services using the Markov decision process, without prior knowledge of the user's mobility pattern. The authors aimed to reduce migration time.

Ilager et al. [3] proposed a deep reinforcement learning model to enable the scheduler to adjust to changing environments through asynchronous updates quickly. This novel idea

achieves lower energy consumption than existing reinforcement learning models, allowing the scheduling of resource-intensive tasks on more powerful machines. A weakness of the proposed approach is the need for continuous profiling of CPU, RAM, disk, and bandwidth demand of new tasks in real edge-cloud environments. This could result in additional computational overhead.

Karimi et al. [19] focus on Mobile Edge Computing, and propose a task scheduling algorithm that relies on deep reinforcement learning and incorporates a resource allocation strategy that involves task migration among edge and central cloud servers. With an appropriate scheduling strategy, they can achieve a shorter response time. However, one drawback of their solution is a reallocation policy, which is necessary since time-sensitive application tasks may arrive at an edge server when resources are insufficient, requiring immediate execution.

Sami et al. [35] model the scheduling problem as an MDP and use deep reinforcement learning techniques to optimize the service placement decision to satisfy QoS requirements. The simulations demonstrate that the proposed DRL-based approach is more effective than heuristic-based solutions for time-sensitive service placement problems. Their main goals are to reduce the number of unfulfilled requests, the number of fog nodes chosen for placement, the number of high-priority services that were not allocated, and the distance of the chosen fog nodes from the users requesting services.

Goudarzi et al. [36] propose a novel approach for addressing the application placement problem in heterogeneous fog computing environments using a distributed deep reinforcement learning technique.

Shi et al. [43] describe a study that uses Deep Reinforcement Learning (DRL) techniques applied to Vehicle-to-Vehicle communication to optimize the utility of scheduling tasks and minimize delay in a fog computing environment that supports vehicle heterogeneity.

Kumar et al. [41] aims to decrease energy consumption and latency in heterogeneous fog environments by proposing a binary offloading strategy based on deep learning that leverages several parallel deep neural networks (DNNs) to facilitate the scheduling decision-making process. After the decision outputs are stored in a relay memory, they are used to train and test all neural networks. Notably, in real-time scheduling, their proposal may require the continuous allocation of resources. Additionally, in a dynamic fog environment where user mobility is critical, their strategy may not be a viable option for data offloading patterns.

Qian et al. [5] formulated a two-step decision problem in cloud–edge based group-distributed manufacturing systems. The scope of the traditional task scheduling problem has been expanded in this paper to include deadline constraints. The authors approach the task placement problem in two steps: first, they prioritize tasks, and second, they schedule the tasks based on their priorities. The proposed approach based on deep reinforcement learning aims to minimize the total delay time of task scheduling by employing two agents: one for prioritizing tasks and another for selecting the suitable node. One challenge that remains to be addressed is the reduction of model training time.

Gazori et al. [15] focused on addressing task scheduling to reduce latency and computing costs while also meeting resource and deadline constraints. To tackle this problem, the authors proposed using either machine learning-based reinforcement learning algorithms or rule-based heuristic algorithms. They suggested that deep reinforcement learning could improve decision-making in resource-constrained environments and formulated a solution using the Markov Decision Process framework. Their proposed solution was a Double Deep Q-Learning (DDQL)-based scheduling algorithm that determines whether to assign a task to a fog node or to a cloud data center. While the DDQL-based scheduling algorithm yielded significant results in simulations, there is still a need to enhance the efficiency and effectiveness of the agents' learning process for this novel idea.

7.6. Fuzzy Logic-Based Scheduling Algorithms

Parimi et al. [30] proposed approach involves assigning tasks in the fog layer to the nodes with the least load, leading to a significant decrease in the waiting time required to complete these tasks. The decision-making algorithm applies task scheduling to fog and cloud resources based on fuzzy logic. Furthermore, the study overlooked the essential aspect of energy consumption.

An alternative task scheduling approach was introduced, where tasks are selected from a set and transferred to a fog node in a sequential list [71]. This study uses a fuzzy logic-based scheduling method that guarantees task deadlines. The priority values determine the degree of urgency for executing the tasks. However, the study did not consider energy consumption and costs. The authors also overlooked the potential benefits of using cloud computing to reduce system costs.

Esseghir et al. [51] examine how user preferences are affected by the constraints of fog nodes and prioritize the reduction of delay and energy consumption. They propose an approach to task scheduling that uses ranked lists and considers user preferences and fog node constraints. The method utilizes a linguistic and fuzzy quantified proposition to prioritize the fog nodes.

Buyya et al. [75] present a QoS-aware approach to placing applications that prioritize the different location requirements for applications, considering the multiple users' expectations. The proposed method evaluates fog instance capabilities based on the current state. Fuzzy logic determines the expected rating for application placement requirements, considering multiple user expectation parameters and the capability rating of fog instances, which is determined by their various state parameters. They aim to maximize the overall gain in Quality of Experience (QoE) for the user by prioritizing a less congested network, ensuring sufficient allocation of resources, and minimizing the time taken to process applications. The authors did not take into account the energy consumption of fog nodes.

8. Research Gaps and Challenges

In what follows, we discuss research gaps and challenges, organizing them along the dimension used for mapping the literature, i.e., optimization criteria and problem solution techniques.

8.1. Optimization Criteria

Producing scheduling decisions that balance contrasting goals like cost, energy consumption, and time (latency or execution time or response time or Completion time) is a challenge. Only 16% of studies consider the above-mentioned optimization criteria, and other works focused on cost and time or energy consumption and cost. Finally, there are research works that base their scheduling decision only on a single optimization criterion (c.f Table 3). Hence, research efforts should focus on multi-objective scheduling algorithms.

An important gap to fill is the study of availability-aware scheduling algorithms. Fog nodes are typically unreliable because they could lose network connectivity, because they can fail, or simply because they can decide to quit a fog colony. Despite this intrinsic unreliability, very few works (about 5%) investigate how to maximize availability, and no one proposes solutions for trading off high availability goals with the other optimization criteria (c.f Table 3).

Another research challenge is to design mobility-aware scheduling algorithms accounting for the dynamic movement of devices within a fog environment, where their locations, connectivity, and processing capabilities are subject to change [14]. To overcome this obstacle, algorithms and strategies must be implemented to handle device mobility efficiently. Tasks can be assigned to the most appropriate resources while accommodating changing network topologies and the impact on real-time application requirements. Mobility-aware scheduling algorithms are essential for maintaining the reliability and efficiency of fog computing systems, especially when device movement and changing conditions are involved.

8.2. Problem Solution Techniques

The literature analysis shows that the research community largely adopts machine-learning solutions to solve the task scheduling problem. In this area, a challenge and gap to fill is the design of explainable ML models that allow the verification of the scheduling decision produced.

Another challenge is the design of autonomous task scheduling algorithms. This challenge involves creating intelligent and responsive scheduling algorithms that can operate effectively and autonomously adapt their decisions in dynamic, real-time fog computing environments, ensuring that tasks are executed efficiently and meet application-specific needs [64]. This includes factors such as variations in device resources, network conditions, task workloads, and application priorities. Self-adaptive schedulers continuously monitor the environment and can dynamically reconfigure task assignments, priorities, and resource allocations.

9. Conclusions

In summary, our exploration of the fog computing domain's task scheduling landscape reveals a notable evolution marked by substantial expansion and transformative changes. Through an extensive literature review, we have uncovered crucial insights into the challenges, solutions, and future prospects of task scheduling in fog computing.

At the heart of fog task scheduling lies the persistent pursuit of efficiency, encompassing resource allocation, low-latency delivery, energy consumption, and cost-effectiveness. Our analysis underscores the dynamic nature of this field and the ongoing efforts to optimize scheduling processes.

Significantly, our review identifies several critical research gaps that signal a pressing need for further investigation. These gaps include the exploration of AI-driven scheduling using explainable ML techniques, the development of innovative autonomous task scheduling algorithms, and the consideration of mobility-aware, availability-aware, and multi-criteria task scheduling.

These key takeaways emphasize the imperative for continuous advancement in fog computing task scheduling to fully unlock its potential across diverse domains. By addressing these challenges and gaps, the fog computing community can pave the way for enhanced efficiency, reliability, and adaptability in task scheduling, contributing to the continued growth and impact of fog computing.

Author Contributions: Conceptualization, J.M. and E.C.; Methodology, J.M. and E.C.; Validation, J.M. and E.C.; Formal Analysis, J.M. and E.C.; Investigation, J.M. and E.C.; Resources, J.M. and E.C.; Data Curation, J.M. and E.C.; Writing—original Draft Preparation, J.M. and E.C.; Writing—review and Editing, J.M. and E.C.; Visualization, J.M. and E.C.; Supervision, E.C.; Funding Acquisition, E.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by DRONES AS A SERVICE for FIRST EMERGENCY RESPONSE (Ateneo 2021). Grant number is RG12117A8AFA07F7.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to ethical considerations and to protect participant privacy.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rahman, G.; Chuah, C.W. Fog Computing, Applications, Security and Challenges, Review. *Int. J. Eng. Technol.* **2018**, *7*, 1615. [CrossRef]
2. Aslanpour, M.S.; Gill, S.S.; Toosi, A. Performance Evaluation Metrics for Cloud, Fog and Edge Computing: A Review, Taxonomy, Benchmarks and Standards for Future Research. *Internet Things* **2020**, *12*, 100273. [CrossRef]
3. Tuli, S.; Ilager, S.; Ramamohanarao, K.; Buyya, R. Dynamic Scheduling for Stochastic Edge-Cloud Computing Environments Using A3C Learning and Residual Recurrent Neural Networks. *IEEE Trans. Mob. Comput.* **2022**, *21*, 940–954. [CrossRef]

4. Abdel-Kader, R.; El Sayad, N.; Rizk, R. Efficient energy and completion time for dependent task computation offloading algorithm in industry 4.0. *PLoS ONE* **2021**, *16*, e0252756. [CrossRef] [PubMed]
5. Xiong, J.; Guo, P.; Wang, Y.; Meng, X.; Zhang, J.; Qian, L.; Yu, Z. Multi-agent deep reinforcement learning for task offloading in group distributed manufacturing systems. *Eng. Appl. Artif. Intell.* **2023**, *118*, 105710. [CrossRef]
6. Nikolopoulos, V.; Nikolaidou, M.; Voreakou, M.; Anagnostopoulos, D. Fog Node Self-Control Middleware: Enhancing context awareness towards autonomous decision making in Fog Colonies. *Internet Things* **2022**, *19*, 100549. [CrossRef]
7. Apat, H.K.; Sahoo, B.; Maiti, P. Service Placement in Fog Computing Environment. In Proceedings of the 2018 International Conference on Information Technology (ICIT), Bhubaneswar, India, 19–21 December 2018; pp. 272–277. [CrossRef]
8. Dehury, C.; Srirama, S. Personalized Service Delivery using Reinforcement Learning in Fog and Cloud Environment. In Proceedings of the iiWAS2019: The 21st International Conference on Information Integration and Web-based Applications & Services, Munich, Germany, 2–4 December 2019; pp. 522–529. [CrossRef]
9. Wang, J.; Hu, J.; Min, G.; Zomaya, A.Y.; Georgalas, N. Fast Adaptive Task Offloading in Edge Computing Based on Meta Reinforcement Learning. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 242–253. [CrossRef]
10. Raju, M.R.; Mothku, S.K. Delay and energy aware task scheduling mechanism for fog-enabled IoT applications: A reinforcement learning approach. *Comput. Netw.* **2023**, *224*, 109603. [CrossRef]
11. Lima, D.; Miranda, H. A geographical-aware state deployment service for Fog Computing. *Comput. Netw.* **2022**, *216*, 109208. [CrossRef]
12. Wu, H.Y.; Lee, C.R. Energy Efficient Scheduling for Heterogeneous Fog Computing Architectures. In Proceedings of the 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, 23–27 July 2018; Volume 1, pp. 555–560. [CrossRef]
13. Das, R.; Inuwa, M.M. A review on fog computing: Issues, characteristics, challenges, and potential applications. *Telemat. Inform. Rep.* **2023**, *10*, 100049. [CrossRef]
14. Waqas, M.; Niu, Y.; He, J.; Ahmed, M.; Chen, X.; Li, Y.; Jin, D.; Han, Z. Mobility-Aware Fog Computing in Dynamic Environments: Understandings and Implementation. *IEEE Access* **2018**, *7*, 38867–38879. [CrossRef]
15. Gazori, P.; Rahbari, D.; Nickray, M. Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach. *Future Gener. Comput. Syst.* **2019**, *110*, 1098–1115. [CrossRef]
16. Zhang, C.; Zheng, Z. Task migration for mobile edge computing using deep reinforcement learning. *Future Gener. Comput. Syst.* **2019**, *96*, 111–118. [CrossRef]
17. Mai, L.; Dao, N.N.; Park, M. Real-Time Task Assignment Approach Leveraging Reinforcement Learning with Evolution Strategies for Long-Term Latency Minimization in Fog Computing. *Sensors* **2018**, *18*, 2830. [CrossRef] [PubMed]
18. Wang, J.; Li, D. Task Scheduling Based on a Hybrid Heuristic Algorithm for Smart Production Line with Fog Computing. *Sensors* **2019**, *19*, 1023. [CrossRef] [PubMed]
19. Karimi, E.; Chen, Y.; Akbari, B. Task offloading in vehicular edge computing networks via deep reinforcement learning. *Comput. Commun.* **2022**, *189*, 193–204. [CrossRef]
20. Bukhari, M.; Ghazal, T.; Abbas, S.; Khan, M.; Farooq, U.; Wahbah, H.; Ahmad, M.; Khan, M. An Intelligent Proposed Model for Task Offloading in Fog-Cloud Collaboration Using Logistics Regression. *Comput. Intell. Neurosci.* **2022**, *2022*, 1–25. [CrossRef]
21. Yin, L.; Luo, J.; Luo, H. Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4712–4721. [CrossRef]
22. Goudarzi, M.; Palaniswami, M.; Buyya, R. Scheduling IoT Applications in Edge and Fog Computing Environments: A Taxonomy and Future Directions. *ACM Comput. Surv.* **2022**, *55*, 1–41. [CrossRef]
23. Mahmud, M.; Ramamohanarao, K.; Buyya, R. Application Management in Fog Computing Environments: A Taxonomy, Review and Future Directions. *ACM Comput. Surv.* **2020**, *53*, 1–43. [CrossRef]
24. Singh, J.; Singh, P.; Amhoud, E.M.; Hedabou, M. Energy-Efficient and Secure Load Balancing Technique for SDN-Enabled Fog Computing. *Sustainability* **2022**, *14*, 12951. [CrossRef]
25. Mohamed, I.; Al-Mahdi, H.; Tahoun, M.; Nassar, H. Characterization of task response time in fog enabled networks using queueing theory under different virtualization modes. *J. Cloud Comput.* **2022**, *11*, 21. [CrossRef]
26. Vakilian, S.; Moravvej, S.V.; Fanian, A. Using the Cuckoo Algorithm to Optimizing the Response Time and Energy Consumption Cost of Fog Nodes by Considering Collaboration in the Fog Layer. In Proceedings of the 2021 5th International Conference on Internet of Things and Applications (IoT), Isfahan, Iran, 19–20 May 2021; pp. 1–5. [CrossRef]
27. Razaq, M.; Rahim, S.; Tak, B.; Peng, L. Fragmented Task Scheduling for Load-Balanced Fog Computing Based on Q-Learning. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 4218696. [CrossRef]
28. Movahedi, Z.; Defude, B.; Hosseininia, A.M. An Efficient Population-Based Multi-Objective Task Scheduling Approach in Fog Computing Systems. *J. Cloud Comput.* **2021**, *10*, 53. [CrossRef]
29. Lin, K.; Pankaj, S.; Wang, D. Task offloading and resource allocation for edge-of-things computing on smart healthcare systems. *Comput. Electr. Eng.* **2018**, *72*, 348–360. [CrossRef]
30. Ali, H.; Rout, R.; Parimi, P.; Das, S. Real-Time Task Scheduling in Fog-Cloud Computing Framework for IoT Applications: A Fuzzy Logic based Approach. In Proceedings of the 2021 International Conference on COMmunication Systems & NETworkS (COMSNETS), Bangalore, India, 5–9 January 2021; pp. 556–564. [CrossRef]
31. Guevara, J.; Fonseca, N. Task scheduling in cloud-fog computing systems. *Peer-Netw. Appl.* **2021**, *14*, 962–977. [CrossRef]

32. Tsai, J.F.; Huang, C.H.; Lin, M.H. An Optimal Task Assignment Strategy in Cloud-Fog Computing Environment. *Appl. Sci.* **2021**, *11*, 1909. [CrossRef]
33. Nguyen, B.M.; Thi Thanh Binh, H.; The Anh, T.; Bao Son, D. Evolutionary Algorithms to Optimize Task Scheduling Problem for the IoT Based Bag-of-Tasks Application in Cloud–Fog Computing Environment. *Appl. Sci.* **2019**, *9*, 1730. [CrossRef]
34. Ghobaei-Arani, M.; Souri, A.; Safara, F.; Norouzi, M. An Efficient Task Scheduling Approach Using Moth-Flame Optimization Algorithm for Cyber-Physical System Applications in Fog Computing. *Trans. Emerg. Telecommun. Technol.* **2020**, *31*, e3770. [CrossRef]
35. Sami, H.; Mourad, A.; Otrok, H.; Bentahar, J. Demand-Driven Deep Reinforcement Learning for Scalable Fog and Service Placement. *IEEE Trans. Serv. Comput.* **2022**, *15*, 2671–2684. [CrossRef]
36. Goudarzi, M.; Palaniswami, M.; Buyya, R. A Distributed Deep Reinforcement Learning Technique for Application Placement in Edge and Fog Computing Environments. *IEEE Trans. Mob. Comput.* **2021**, *22*, 2491–2505. [CrossRef]
37. Jazayeri, F.; Shahidinejad, A.; Ghobaei-Arani, M. Autonomous computation offloading and auto-scaling the in the mobile fog computing: A deep reinforcement learning-based approach. *J. Ambient Intell. Humaniz. Comput.* **2021**, *12*, 8265–8284. [CrossRef]
38. Abdulredha, M.N.; Attea, B.A.; Jabir, A.J. An Evolutionary Algorithm for Task scheduling Problem in the Cloud-Fog environment. *J. Phys. Conf. Ser.* **2021**, *1963*, 012044. [CrossRef]
39. Yang, Y.; Zhao, S.; Zhang, W.; Chen, Y.; Luo, X.; Wang, J. DEBTS: Delay Energy Balanced Task Scheduling in Homogeneous Fog Networks. *IEEE Internet Things J.* **2018**, *5*, 2094–2106. [CrossRef]
40. Nikoui, T.S.; Balador, A.; Rahmani, A.M.; Bakhshi, Z. Cost-Aware Task Scheduling in Fog-Cloud Environment. In Proceedings of the 2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST), Tehran, Iran, 10–11 June 2020; pp. 1–8. [CrossRef]
41. Sarkar, I.; Kumar, S. Deep Learning-Based Energy-Efficient Computational Offloading Strategy in Heterogeneous Fog Computing Networks. *J. Supercomput.* **2022**, *78*, 15089–15106. [CrossRef]
42. Fan, J.; Ma, R.; Gao, Y.; Gu, Z. A Reinforcement Learning based Computing Offloading and Resource Allocation Scheme in F-RAN. *EURASIP J. Adv. Signal Process.* **2021**, *2021*, 91. [CrossRef]
43. Shi, J.; Du, J.; Wang, J.; Yuan, J. Deep Reinforcement Learning-Based V2V Partial Computation Offloading in Vehicular Fog Computing. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference (WCNC), Nanjing, China, 29 March–1 April 2021; pp. 1–6. [CrossRef]
44. Jamil, B.; Shojafar, M.; Ahmed, I.; Ullah, A.; Munir, K.; Ijaz, H. A Job Scheduling Algorithm for Delay and Performance Optimization in Fog Computing. *Concurr. Comput. Pract. Exp.* **2019**, *32*, e5581. [CrossRef]
45. Yang, M.; Zhu, H.; Wang, H.; Koucheryavy, Y.; Samouylov, K.; Qian, H. An Online Learning Approach to Computation Offloading in Dynamic Fog Networks. *IEEE Internet Things J.* **2020**, *8*, 1572–1584. [CrossRef]
46. Alatoun, K.; Matrouk, K.; Mohammed, M.A.; Nedoma, J.; Martinek, R.; Zmij, P. A Novel Low-Latency and Energy-Efficient Task Scheduling Framework for Internet of Medical Things in an Edge Fog Cloud System. *Sensors* **2022**, *22*, 5327. [CrossRef]
47. Wu, B.; Lv, X.; Deyah Shamsi, W.; Gholami Dizicheh, E. Optimal deploying IoT services on the fog computing: A metaheuristic-based multi-objective approach. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 10010–10027. [CrossRef]
48. Baek, J.Y.; Kaddoum, G. Heterogeneous Task Offloading and Resource Allocations via Deep Recurrent Reinforcement Learning in Partial Observable Multi-Fog Networks. *IEEE Internet Things J.* **2020**, *8*, 1041–1056. [CrossRef]
49. Wang, Y.; Huang, H.; Miyazaki, T.; Guo, S. Traffic and Computation Co-Offloading With Reinforcement Learning in Fog Computing for Industrial Applications. *IEEE Trans. Ind. Inform.* **2018**, *15*, 976–986. [CrossRef]
50. Wang, S.; Hu, Z.; Deng, Y.; Hu, L. Game-Theory-Based Task Offloading and Resource Scheduling in Cloud-Edge Collaborative Systems. *Appl. Sci.* **2022**, *12*, 6154. [CrossRef]
51. Benblidia, M.; Brik, B.; Merghem-Boulahia, L.; Esseghir, M. Ranking Fog nodes for Tasks Scheduling in Fog-Cloud Environments: A Fuzzy Logic Approach. In Proceedings of the 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), Tangier, Morocco, 24–28 June 2019; pp. 1451–1457. [CrossRef]
52. Aburukba, R.O.; AliKarrar, M.; Landolsi, T.; El-Fakih, K. Scheduling Internet of Things requests to minimize latency in hybrid Fog–Cloud computing. *Future Gener. Comput. Syst.* **2020**, *111*, 539–551. [CrossRef]
53. Canali, C.; Lancellotti, R. GASP: Genetic Algorithms for Service Placement in Fog Computing Systems. *Algorithms* **2019**, *12*, 201. [CrossRef]
54. Zhang, J.; Guo, H.; Liu, J. *A Reinforcement Learning Based Task Offloading Scheme for Vehicular Edge Computing Network*; Artificial Intelligence for Communications and Networks; Springer: Harbin, China, 2019; pp. 438–449. [CrossRef]
55. Ren, Y.; Sun, Y.; Peng, M. Deep Reinforcement Learning Based Computation Offloading in Fog Enabled Industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2021**, *17*, 4978–4987. [CrossRef]
56. Ahvar, E.; Ahvar, S.; Mann, Z.; Crespi, N.; Glitho, R.; Garcia-Alfaro, J. DECA: A Dynamic Energy Cost and Carbon Emission-Efficient Application Placement Method for Edge Clouds. *IEEE Access* **2021**, *9*, 70192–70213. [CrossRef]
57. Bozorgchenani, A.; Disabato, S.; Tarchi, D.; Roveri, M. An energy harvesting solution for computation offloading in Fog Computing networks. *Comput. Commun.* **2020**, *160*, 577–587. [CrossRef]
58. Hosseinioun, P.; Kheirabadi, M.; Kamel Tabbakh, S.R.; Ghaemi, R. A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm. *J. Parallel Distrib. Comput.* **2020**, *143*, 88–96. [CrossRef]

59. Vanhala, E.; Kasurinen, J.; Knutas, A.; Herala, A. The Application Domains of Systematic Mapping Studies: A Mapping Study of the First Decade of Practice With the Method. *IEEE Access* **2022**, *10*, 37924–37937. [CrossRef]
60. Giuffrida, R.; Dittrich, Y. Empirical studies on the use of social software in global software development – A systematic mapping study. *Inf. Softw. Technol.* **2013**, *55*, 1143–1164. [CrossRef]
61. Wohlin, C. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE '14), New York, NY, USA, 13–14 May 2014. [CrossRef]
62. Rožanc, I.; Mernik, M. The screening phase in systematic reviews: Can we speed up the process? *Adv. Comput.* **2021**, *123*, 115–191. [CrossRef]
63. Zhang, H.; Babar, M.A.; Tell, P. Identifying Relevant Studies in Software Engineering. *Inf. Softw. Technol.* **2011**, *53*, 625–637. [CrossRef]
64. Pandit, M.K.; Mir, R.; Chishti, M.A. Adaptive task scheduling in IoT using reinforcement learning. *Int. J. Intell. Comput. Cybern.* **2020**. [CrossRef]
65. Sopin, E.; Nikita, Z.; Ageev, K.; Shorgin, S. *Analysis of the Response Time Characteristics of the Fog Computing Enabled Real-Time Mobile Applications*; Springer: St. Petersburg, Russia, 2020; pp. 99–109. [CrossRef]
66. Gill, S.S.; Chana, I.; Singh, M.; Buyya, R. CHOPPER: An intelligent QoS-aware autonomic resource management approach for cloud computing. *Clust. Comput.* **2018**, *21*, 1203–1241. [CrossRef]
67. Hosseinioun, P.; Kheirabadi, M.; Kamel Tabbakh, S.R.; Ghaemi, R. ATask Scheduling Approaches in Fog Computing: A Survey. *Trans. Emerg. Telecommun. Technol.* **2022**, *33*. [CrossRef]
68. Aburukba, R.; Landolsi, T.; Omer, D. A heuristic scheduling approach for fog-cloud computing environment with stationary IoT devices. *J. Netw. Comput. Appl.* **2021**, *180*, 102994. [CrossRef]
69. Khaleel, M. Hybrid cloud-fog computing workflow application placement: Joint consideration of reliability and time credibility. *Multimed. Tools Appl.* **2022**, *82*, 1–32. [CrossRef]
70. Poltronieri, F.; Tortonesi, M.; Stefanelli, C.; Suri, N. Reinforcement Learning for value-based Placement of Fog Services. In Proceedings of the 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), Bordeaux, France, 17–21 May 2021; pp. 466–472.
71. Rai, K.; Vemireddy, S.; Rout, R. Fuzzy Logic based Task Scheduling Algorithm in Vehicular Fog Computing Framework. In Proceedings of the 2021 IEEE 18th India Council International Conference (INDICON), Guwahati, India, 19–21 December 2021; pp. 1–6. [CrossRef]
72. Wu, Q.; Wu, Z.; Zhuang, Y.; Cheng, Y. Adaptive DAG Tasks Scheduling with Deep Reinforcement Learning. In Proceedings of the 18th International Conference, ICA3PP 2018, Guangzhou, China, 15–17 November 2018; Proceedings, Part II; pp. 477–490. [CrossRef]
73. Kumar, M.S.; Karri, G.R. EEOA: Cost and Energy Efficient Task Scheduling in a Cloud-Fog Framework. *Sensors* **2023**, *23*, 2445. [CrossRef]
74. Farhat, P.; Sami, H.; Mourad, A. Reinforcement R-learning model for time scheduling of on-demand fog placement. *J. Supercomput.* **2020**, *76*, 1–23. [CrossRef]
75. Mahmud, M.; Srirama, S.; Ramamohanarao, K.; Buyya, R. Quality of Experience (QoE)-aware placement of applications in Fog computing environments. *J. Parallel Distrib. Comput.* **2018**, *132*, 190–203. [CrossRef]