electronics

MDPI

Review

A State-of-the-Art Review of Task Scheduling for Edge Computing: A Delay-Sensitive Application Perspective

Amin Avan *, Akramul Azim and Qusay H. Mahmoud

Department of Electrical, Computer and Software Engineering, Ontario Tech University, Oshawa, ON L1G 0C5, Canada; akramul.azim@ontariotechu.ca (A.A.); qusay.mahmoud@ontariotechu.ca (Q.H.M.)
* Correspondence: amin.avan@ontariotechu.net

Abstract: The edge computing paradigm enables mobile devices with limited memory and processing power to execute delay-sensitive, compute-intensive, and bandwidth-intensive applications on the network by bringing the computational power and storage capacity closer to end users. Edge computing comprises heterogeneous computing platforms with resource constraints that are geographically distributed all over the network. As users are mobile and applications change over time, identifying an optimal task scheduling method is a complex multi-objective optimization problem that is NP-hard, meaning the exhaustive search with a time complexity that grows exponentially can solve the problem. Therefore, various approaches are utilized to discover a good solution for scheduling the tasks within a reasonable time complexity, while achieving the most optimal solution takes exponential time. This study reviews task scheduling algorithms based on centralized and distributed methods in a three-layer computing architecture to identify their strengths and limitations in scheduling tasks to edge service nodes.

Keywords: edge computing; internet of things; task scheduling

Citation: Avan, A.; Azim, A.; Mahmoud, Q.H. A State-of-the-Art Review of Task Scheduling for Edge Computing: A Delay-Sensitive Application Perspective. *Electronics* **2023**, *12*, 2599. https://doi.org/10.3390/electronics12122599

Academic Editors: Charalabos Skianis, Philippe Krief, Enric Pages Montanera and John Soldatos

Received: 10 May 2023
Revised: 31 May 2023
Accepted: 6 June 2023
Published: 8 June 2023

1. Introduction

Offloading compute- and memory-intensive tasks to a remote server decreases power consumption and increases the battery lifetime of end-user devices; thus, cloud computing was provided via the mobile operator and the Internet to support users' needs for computation power and data. Cloud computing was a reasonable solution to the demand for computations and data rates in the cloud era when mobile users and IoT devices only consumed data, such as watching videos on mobile phones while sending data such as sending an email. However, cloud computing faces significant challenges in meeting the needs of specific IoT applications. For example, applications that require real-time data processing, such as autonomous vehicles, can experience latency when processing data in the cloud. In addition, applications with bandwidth-intensive requirements, such as surveillance cameras, cannot accurately transfer data to the cloud for processing. Moreover, applications that deal with sensitive data, such as healthcare and financial systems, require high levels of security and privacy, which can be compromised when data are transferred to the cloud. Finally, some applications need cost-efficient computation and storage, which may not be achievable with cloud computing. In addition, delays and jitters provided by long wide area networks (WANs) are considerable obstacles in cloud computing for data transmission in the user-application interaction, while delay controlling in the WAN scale is problematic [1]. Furthermore, current mobile users and IoT devices applications and services are highly demanding in data, computation, and quality of services (QoS); the current mobile users and IoT devices create data as much as or even more than the amount of data they consume with the real-time response need [2]. For instance, many mobile users and IoT devices need a lot of bandwidth for uploading and sharing high-quality videos on social

network platforms. Additionally, recent cameras are artificial-intelligence-powered cameras with real-time object detection, face recognition, and semantic segmentation abilities that need intensive computation and low latency [3]. However, the QoS of the mentioned application exponentially decreased as much as the latency increased [3]. According to Cisco and IHS Markit predictions [4,5], the number of connected IoT devices will exceed 100 billion in 2030 from around 26 billion in 2020. In addition to the number of devices, another significant challenge of the network is the management of both computation and data communication in the post-cloud era [6]. The end-user devices generate abundant data in the post-cloud era, and the data need to be processed for diverse applications such as healthcare IoT, vehicular IoT, autonomous driving, unmanned aerial vehicles, satellite IoT, and industrial IoT in wireless networks [7]. However, edge computing can bring computation and storage power near to the user instead of the cloud; thus, edge computing can considerably help the flow of backbone networks, improve end-to-end communication speed, and positively contribute to the quality of experience (QoE) for the users [8]. Moreover, since edge computing provides computing and storage resources close to the end user, edge computing enables users to execute highly demanding applications that are sensitive to delay. Finally, edge computing is beneficial and necessary for modern IoT applications as it can securely process sensitive data, enable large-scale data processing, facilitate real-time data streaming, enable fast response times in smart applications, improve application accessibility in remote areas, and decrease internet bandwidth utilization.

Edge computing systems encounter an optimization problem that involves the coordination of wireless frequency spectrum, computing, and storage resources while considering power consumption and latency limitations. Edge services, including sensing, communication, computing, storage, and intelligence, could be provided for the users through base stations, access points, and roadside units. Furthermore, one of the fifth-generation of mobile communications (5G) goals is to operate for many mobile users and IoT devices with different applications and intense processing and storage requirements [9]. Consequently, with the advent of the 5G networks and drastic increment in the number of users and their intense requirements, the network computing paradigm needs to change from centralized data centers to decentralized near the edge of the network where the data are generated and consumed. In addition to intense computation and data transmission in 5G, the latency harms interactive response despite adequate bandwidth. In addition, data transmission is one of the critical elements in the cloud computing paradigm; however, data transmission encounters several challenges, such as massive data volume, communication cost, reliability, data privacy and security, and data administrative policies. Moreover, cloud computing suffers from end-to-end latency for highly interactive applications such as visualization, raster graphics editor, augmented reality (AR), and virtual reality (VR) [10]. Therefore, the network encounters massive amounts of generated data and needs to manage the network resources for the deployed applications in the network. Furthermore, some of the applications in healthcare, autonomous vehicle, and industrial domains deployed in the network have one or all the following characteristics: (1) they require short or reliable response time; (2) they have private data; and (3) they generate tremendous quantities of data. For instance, connected and autonomous driving (CAD) is an application with real-time requirements for avoiding collision and individuals' safety; thus, simultaneous localization and mapping (SLAM) is vital. However, SLAM is a time-sensitive and high-computation operation that is needed to be executed by a server. Although the cloud can quickly perform tasks, communication latency to the cloud is not suitable for the SLAM. Nonetheless, the edge can satisfy both the computation and delay demands of SLAM [11]. In addition to CAD, industrial IoT services need reliable response time for the applications such as real-time controlling actions [12]. Video surveillance is another application that, if it is developed enough, a jumbo volume of data will generate; thus, the cloud is an inefficient solution for video analysis due to the communication delay for the data transmission [13].

Furthermore, edge computing can address healthcare IoTs computation and storage concerns. Since individuals who encounter chronic or critical diseases are monitored

continuously, their data need to be processed and analyzed in real time while the data volume is enormous. Therefore, edge computing can execute the patient data near where it is generated, which leads to eliminating communication delays, mitigating traffic collisions, and preventing data breaches in the network [14].

Consequently, offloading the computation of the mentioned applications to the cloud raises safety, reliability, and communication concern. Therefore, edge computing brings computation power near applications with intense computation, privacy, and safety concerns. Edge computing enables several real-time applications, such as telemedicine, haptic telecommunications, and immersive VR services, which have strict real-time requirements [15].

In addition to the distance (geographical distance between a client and a data center) that contributes to the increment of latency in cloud computing, any improvement in security, energy efficiency, and manageability of WAN leads to higher latency in network response. For instance, firewalls, authentications, transmission medium (link), network device functions, and energy-aware techniques in wireless communication are employed to improve the mentioned WAN goals while increasing the network latency.

One of the initial steps in the evolution from cloud computing to edge computing was the development of cloudlets. Cloudlets are designed to offer computation and storage resources at specific locations with high demand for these resources [16]. Although cloudlet is accessible via a Wi-Fi connection, cloudlet is not part of the cloud computing network, and the end-user needs to switch from the network (cloud computing) to Wi-Fi (cloudlet). Nevertheless, edge computing aims to provide resources near users along with cloud computing coverage, which can support users' mobility throughout the network.

Many papers have reviewed the progress and development of edge computing in recent years. In this paper, we examine the state-of-the-art surveys on scheduling algorithms in edge computing and compare them based on specific criteria, as presented in Table 1. We evaluate the suitability of task scheduling algorithms for time-sensitive applications and systems, as indicated in Table 1.

Table 1. Summary of recent surveys and comparison with this review.

Ref.	Edge Computing	Resource and Task Management	Real-Time Perspective	Summary
[17]	Yes	No	No	Presents a classification of industrial aspects benefiting from IoT and edge computing. Proposes two real-world use cases that address urban smart living challenges and proposes a new architecture based on edge-IoT for e-healthcare.
[18]	Yes	No	No	Explains the issues in the collaboration between edge computing and CPS, reviewing recent papers focusing on and classifying QoS optimization.
[14]	Yes	No	No	Reviews the research on the collaboration between edge computing and healthcare applications, focusing on architecture and techniques. Discusses the challenges of healthcare applications in edge computing and provide an overview of all data operations.
[6]	Yes	No	No	Investigates edge computing as a next-generation computing technology. Elaborates on how edge computing can reduce operating costs and enhance security. Analyzes the aspects of data transmission and communication within edge computing.
[1]	Yes	No	No	Overview of edge computing architecture, applications, and security includes the analysis of potential security risks and vulnerabilities. Several protection methods are explored to mitigate security threats.

Table 1. *Cont.*

Ref.	Edge Computing	Resource and Task Management	Real-Time Perspective	Summary
[19]	Yes	No	No	Examines virtualization techniques in computation and networking resources and explore their deployment in edge computing. Investigates the relationship between virtualization techniques and the requirements of IoT services.
[20]	Yes	No	No	Explains the definitions and core characteristics of edge computing and investigates different application scenarios.
[21]	Yes	Yes	No	Research works on task offloading are analyzed from a stochastic perspective, and a taxonomy comprising Markov chains, Markov processes, and hidden Markov models is presented.
[22]	Yes	Yes	No	Reviews recent research on VEC regarding different aspects, presents various VEC applications, and categorize them.
[23]	Yes	Yes	No	Reviews the papers on resource management in edge computing, wherein different aspects of resource management are explained, including computation offloading, resource allocation, and resource provisioning.
[24]	Yes	Yes	No	Examines various task scheduling methods in the context of edge computing and explores the relationship between these methods and their corresponding problem formulations.
[25]	Yes	Yes	No	Reviews resource management methods suitable for cloud, edge, and fog environments. Proposes an assessment framework comprising measurements for resource management algorithms in edge computing.
[26]	Yes	Yes	No	Reviews the research progress made in edge computing regarding the service placement problem (SPP). Categorizes the various methods employed for task scheduling and other aspects associated with SPP.
[2]	Yes	Yes	No	Reviews recent research progress in task offloading techniques for edge computing.
[27]	Yes	Yes	No	Reviews the progress made on energy-aware aspects of edge computing in different domains, including task management.
[23]	Yes	Yes	No	Explains the edge computing architecture and its collaboration with different task scheduling algorithms and classify recent research on resource management in edge computing. Divides the scheduling algorithms based on their operation mode.
[12]	Yes	Yes	No	Explains the collaboration between IIoT and edge computing, as well as the related research progress. Provides a review of the advancements achieved in various technical aspects of edge computing, including task scheduling.
[28]	Yes	Yes	No	Provides an overview of the advancements in computation offloading and categorizes computation offloading models into different classes. Explains the fundamental concepts of computation offloading and discuss various methods utilized in it.

Table 1. *Cont.*

Ref.	Edge Computing	Resource and Task Management	Real-Time Perspective	Summary
[29]	Yes	Yes	No	Provides a taxonomy of recent task scheduling algorithms in edge/fog computing.
[30]	Yes	Yes	No	Reviews the recent research progress of task scheduling algorithms in edge computing, categorizing them based on task dependency and the number of available servers.
This paper	Yes	Yes	Yes	A comprehensive survey examines the recent progress in task scheduling algorithms. The algorithms are categorized based on their operation mode, problem formulation method, and their suitability for time-sensitive applications.

Consequently, the contributions of this review are as follows:

- We elucidate the differences between the operation mode and execution paradigms of edge computing and cloud computing. We analyze each paradigm from multiple aspects, including deployment, distance, latency, computation power, and storage capacity.
- We explain the architecture of edge computing and its collaboration with the end user and the cloud. In addition, we illustrate the network architecture, which encompasses the end-user (things), edge, and cloud components. We also provide an explanation of each layer within this architecture. Moreover, we conduct a comprehensive review of the available computing resources within edge computing. Additionally, we distinguish and outline the distinctive characteristics associated with each resource type. Furthermore, we present advancements in 6G as an emerging technology and consequential impact on edge computing.
- We present a step-by-step explanation of the task scheduling procedure in edge computing and discuss why edge computing is considered a promising approach for offloading time-sensitive and data-sensitive applications.
- We explore the optimization perspectives and objectives presented in state-of-the-art papers on task scheduling and examine how each paper formulates the scheduling problem.
- We categorize the task scheduling techniques into two main categories, distinguished by their operation and execution mode. Moreover, we thoroughly examine each category, presenting a detailed discussion of their characteristics. Additionally, we clarify the advantages and disadvantages inherent in each technique. Furthermore, we construct a table that compares over fifty state-of-the-art works on task scheduling to each other, considering multiple parameters.
- We clarify which task-scheduling techniques appear promising for effectively scheduling time-sensitive applications.

Table 2 provides a comprehensive list of frequently used acronyms in the survey. The rest of this paper is organized as follows. Section 2 discusses the fundamental concepts of task scheduling in edge computing, while Section 3 outlines the research methodology used in this study. Section 4 provides an overview of network architecture in edge computing. Section 5 examines optimization properties and how they can be utilized to improve task scheduling in edge computing. Section 6 discusses the current techniques for solving the task scheduling problem in edge computing. Section 7 focuses on the suitability of different methods for scheduling the tasks of real-time applications. Finally, Section 8 highlights the open issues and future research directions in task scheduling in edge computing.

Table 2. Acronyms used in this survey.

Acronym	Definition
WAN	Wide Area Network
QoS	Quality of Service
IoT	Internet of Things
QoE	Quality of Experience
AR	Augmented Reality
VR	Virtual Reality
CAD	Connected and Autonomous Driving
SLAM	Simultaneous Localization and Mapping
RSU	Roadside Unit
UAV	Unmanned Aerial Vehicle
WBAN	Wireless Body Area Networks
RQ	Research Question
IC	Inclusion Criteria
LTE	Long-Term Evolution
Wi-Fi	Wireless Fidelity
CPU	Central Processing Unit
GPU	Graphics Processing Unit
ASIC	Application Specific Integrated Circuit
FPGA	Field Programmable Gate Array
DAG	Directed Acyclic Graph
XR	Extended Reality
AI	Artificial Intelligence
ML	Machine Learning
ILP	Integer Linear Programming
MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Non-Linear Programming
MDP	Markov Decision Process
ADMM	Alternating Direction Method of Multipliers
EDF	Earliest Deadline First
FCFS	First Come First Serve
NSGA	Non-dominated Sorting Genetic Algorithm
MOWO	Multi-Objective Whale Optimization
SLA	Service Level Agreement
DRL	Deep Reinforcement Learning
ASA	Simulated Annealing Approach
DQN	Deep Q-learning Network
FL	Federated Learning
LSTM	Long Short-Term Memory
MAML	Model-agnostic Meta-learning
IIoT	Industrial Internet of Things
IoV	Internet of Vehicles

2. Task Scheduling in Edge Computing

One significant distinction between cloud computing and edge computing is their computation paradigm, which must be considered when implementing task scheduling methods. Cloud computing is a centralized computing paradigm, whereas edge computing is a distributed computing paradigm. Therefore, to ensure optimal performance, the task scheduling techniques employed in each should be different and compatible with the characteristics of each paradigm. Although cloud computing and edge computing are services for offloading computation tasks from the end-user device to a server, they have different computation paradigms that make them suitable for various applications. Cloud computing is a centralized computation facility that is proper for process-intensive applications. In contrast, the bottleneck of cloud computing is the networks' bandwidth and privacy guarantee for sensitive data. Conversely, edge computing could address concerns such as latency, bandwidth, security, privacy, and limited battery power of mobile devices and embedded systems [7,31,32]. In addition to different computation paradigms

in cloud computing and edge computing, cloud computing services are accessible by the Internet for end-users. However, end-users utilize edge server nodes near them, which can vary in different locations. Therefore, edge computing can support applications that have latency, bandwidth, security, and privacy considerations. Table 3 highlights the technical differences between cloud computing and edge computing [29].

Table 3. General comparison of cloud and edge computing.

Characteristic	Cloud Computing	Edge Computing
Deployment	Centralized	Distributed
Distance	High	Low
Latency	High	Low
Computation power	Ample	Limited
Storage capacity	Ample	Limited

In recent years, mobile applications such as social networks, games, healthcare, mobile payment, VR, and AR are becoming increasingly complex and demand more computing resources and energy [23]. Hence, offloading the possible tasks of the mentioned applications to edge computing can increase the QoS and reduce energy consumption for time-sensitive applications and power-constrained devices, respectively. For example, edge computing enables vehicles to offload the tasks of sensory data analysis and navigation path identification to edge nodes such as roadside units (RSUs) for fast processing [33]. Moreover, unmanned aerial vehicles (UAVs) utilize edge computing, where drones offload their image processing tasks for navigation to the edge servers [34].

Task scheduling generally involves assigning appropriate resources to tasks for execution at the right time. The efficiency metrics of a task scheduling technique increase as it accurately allocates tasks to a suitable server. In edge computing, task scheduling entails participants using a methodology to allocate tasks to edge resources for execution that can be described as follows [23]:

- Participants: the components of the network that collaborate on task execution, such as user, edge, and cloud, are the participants [35].
- Resources: edge computing components that provide a service in the network, such as communication resources, storage resources, caching resources, and computing resources [36].
- Tasks: a unit work in an application in which there are different types of tasks in edge computing such as LiDAR [37] or camera [38] of autonomous vehicles, wireless body area networks (WBAN) of healthcare IoT applications [17].
- Methodology: different methods can be utilized to schedule tasks, including centralized and distributed [39].

Any efficient task scheduling method, either centralized or distributed, needs to consider three aspects [29]:

- Computation offloading: Determining which tasks need to executed by edge computing.
- Resource allocation: Determining which of the edge computing server nodes is the most suitable for the task.
- User mobility: The task scheduling method should regularly check the presence of the end-user as the user might leave or join the covered area.

The network consists of three layers: things, edge, and cloud. The things layer includes various end-user devices, such as smart home devices, surveillance cameras, autonomous vehicles, VR and AR gadgets, unmanned aerial vehicles, and industrial devices in smart factories. The edge and cloud layers host servers that carry out tasks for these end-user devices. The edge servers, situated close to the end users, are suitable for time-sensitive and data-sensitive applications. However, it should be noted that the Edge servers have lower computing power and storage capacity compared to the cloud servers. Accordingly, a task scheduling algorithm considers the availability of servers, including edge servers

and cloud, as well as the tasks of end-users. The algorithm then schedules end-user tasks by considering the requirements of end-users' tasks and the capabilities of edge servers and the cloud. For example, consider Figure 1, which depicts two cars as end-users, three edge servers, and the cloud. Each car wants to offload its entertainment tasks, such as gaming and multimedia player tasks. Now, the scheduling algorithm has two users with specified tasks. The algorithm schedules the tasks based on their computing, memory, and time requirements to edge servers or the cloud. If the tasks are time-sensitive and data-sensitive, they will be scheduled on edge servers. The task scheduling algorithm selects an edge server from all the available edge servers with sufficient computing power and memory capacity to execute the task within the required latency based on the task's requirements. However, if the tasks are compute-intensive and memory-intensive, they will be offloaded to the cloud. The task scheduling algorithm continuously checks for the arrival of new tasks or users to the network to promptly respond and schedule tasks without any delay.

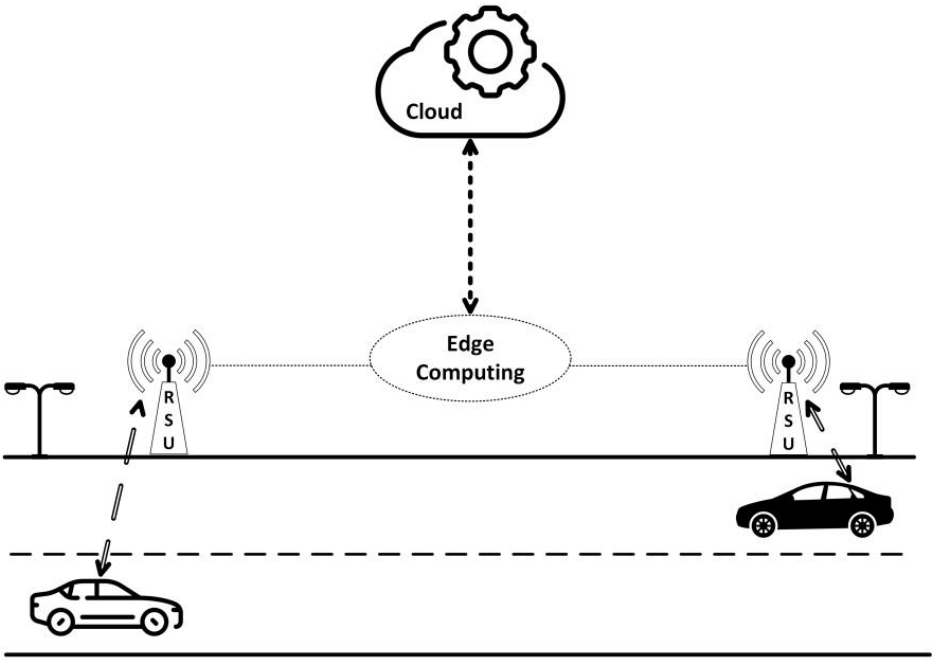

Figure 1. Users, edge, and cloud collaboration.

3. Method

This review paper adheres to the spirit of the guidelines proposed by Kitchenham et al. [40]. This section discusses the methodology used to conduct this review, including the research questions and inclusion criteria considered.

3.1. Research Question

This study intends to answer the following research questions (RQs):

- RQ1: What techniques have been utilized for scheduling the tasks in edge computing?
  - o Through resolving these investigations, a more comprehensive comprehension of different task scheduling methods can be achieved, facilitating an exploration of the advantages and disadvantages of each method to determine appropriate task scheduling approaches for time-sensitive applications.
- RQ2: What techniques are suitable for scheduling the tasks of time-sensitive applications?
  - o Answering this inquiry would clarify which task-scheduling technique is more suitable for time-sensitive applications.

3.2. Inclusion Criteria

Once the research questions of the review have been developed, as mentioned earlier, it is essential to establish appropriate inclusion criteria (IC) for selecting previous studies to be reviewed. This process ensures an unbiased analysis of the primary studies. Therefore, previous studies must fulfill at least one of the following criteria to be included in the review. We review state-of-the-art papers that address various aspects of task scheduling, resource management, resource usage optimization, end-user power, and delay optimization, as well as the implementation of real-time applications and systems in edge computing.

- IC1: Address the task and resource management on edge computing.
- IC2: Address the task scheduling challenge on edge computing.
- IC3: Address the implementation and development of time-sensitive applications in edge computing.
- IC4: Address task complexity in task scheduling to manage various tasks with different levels of complexity.
- IC5: Address resource availability of computational resources, including processors, memory, and storage, at edge devices into the task scheduling algorithm.
- IC6: Address latency requirements of tasks with strict latency requirements to ensure deadlines are met.
- IC7: Address the optimization of energy consumption in edge devices.
- IC8: Address the optimization of computation latency of edge devices in edge computing.
- IC9: Address security and privacy to ensure the confidentiality and integrity of the data being processed on edge devices.
- IC10: Address network bandwidth availability where data need to be transmitted in the edge network.
- IC11: Address users' preferences, such as their desired service quality level or willingness to trade off performance for energy savings.

4. Network Architecture

Although cloud computing encounters difficulties in fulfilling real-time applications' requirements due to the delay, edge computing can only partially handle some enterprise applications such as supply chain management, customer relationship management, large-scale web applications, and large-scale mobile applications. Therefore, a realistic network computing paradigm comprises cloud computing and edge computing as collaborative computation platforms. Consequently, many studies [23,41,42] consider three-tier (three-layered) network architecture composed of the cloud, the edge, and the Things layers to take advantage of the cloud and edge resources. For instance, Fizza et al. [43] proposed two scheduling algorithms for autonomous vehicles that execute tasks based on latency tolerance. Hence, the scheduler executes hard real-time tasks such as braking systems on the local processor, while the firm real-time and soft-real-time tasks are executed in the edge and cloud platforms, respectively. In addition, a meta-heuristic task scheduling algorithm is proposed by [44] to schedule the tasks on either local, edge, or cloud processors by considering the computation power, storage capacity, and bandwidth capability. Consequently, Figure 2 illustrates the three-tier network architecture consisting of the things, edge, and cloud layer adapted from [42].

4.1. Things Layer

Multitudes of different devices with various applications exist in the things layer, including AR gadgets, smart home devices, surveillance cameras, autonomous vehicles, unmanned aerial vehicles, and industrial devices in smart factories. Each device of the thing layer possesses specific amounts of computation and storage capacity, while the device can offload the tasks according to the QoS and QoE requirements. In addition to the processing characteristics, the devices might have a dynamic location; thus, the resource scheduling algorithm needs to consider the mobility attribute of devices.

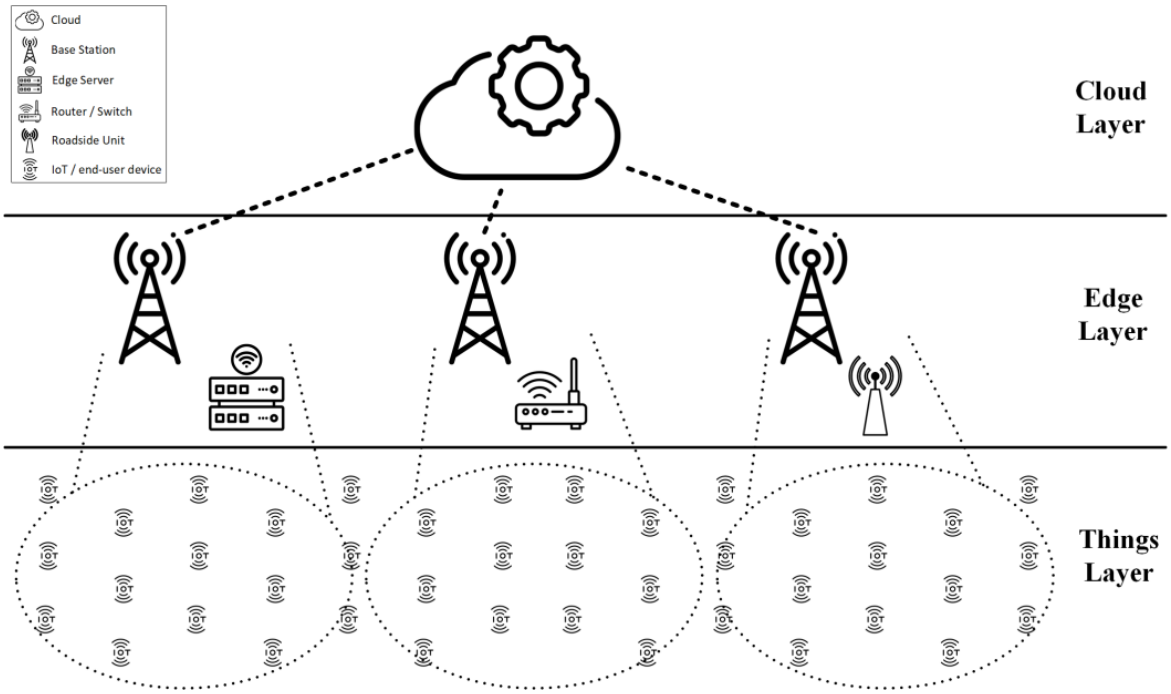

Figure 2. Three-layer network architecture.

4.2. Edge Layer

The edge layer comprises geographically distributed heterogenous nodes such as base stations, edge servers, gateways (e.g., router or switch), and RSUs, which can provide computation and storage capacity near the users. Edge nodes communicate with the Things layer via different wireless access technologies, including Dedicated Short-Range Communications, Wireless Fidelity (Wi-Fi), and Long-Term Evolution (LTE). Edge possesses a mutual connection with cloud and Things layers. Therefore, the edge can be utilized to reduce the communication overhead of the cloud for users to execute users' real-time tasks. Moreover, edge decreases the cloud's bandwidth usage and energy consumption by executing users' tasks that edge nodes can execute. Furthermore, the edge is a proper solution for the cloud to balance its heavy load in specific situations.

4.3. Cloud Layer

As the most powerful processing platform in the network, the cloud provides a pool of computation power and storage capacity. Cloud is a proper platform for deploying, maintaining, developing, updating, and scaling enterprise and large-scale applications. However, the challenge between users and the cloud is (1) the bandwidth limitation due to the geographical distance between the cloud and users and (2) the tremendous number of users that occupy the bandwidth for transmitting tasks and data.

4.4. Network Resources

The network has three different resources: computation, storage, and communication. The computing platform of edge computing is heterogenous, including central processing unit (CPU), graphics processing unit (GPU), application specific integrated circuit (ASIC), and field programmable gate array (FPGA). Although the cloud mainly comprises homogenous computation elements, the edge servers comprise heterogeneous computation platforms such as CPU, GPU, and FPGA. Table 4 exhibits the characteristic of GPU, CPU (ASIC), and FPGA. Consequently, a proper resource scheduling algorithm should consider end-user tasks' parameters and edge computing platforms' characteristics to efficiently exploit the capacity of the computation platform of edge computing for real-time requirements [45].

Table 4. Comparison of GPU, ASIC, FPGA.

Processor	Characteristic
GPU	High latency High power consumption High flexibility
ASIC	Low latency Low power consumption Low flexibility
FPGA	Low latency Low power consumption High flexibility

According to Table 4, FPGA is more power efficient and faster than GPU; also, regarding ASIC, the FPGA is configurable and more flexible; thus, FPGA can be a suitable candidate for computation units of real-time and embedded systems with stringent resource constraints. FPGA is configured with hardware description language (HDL); thus, the kernel of deep learning operations is yielded faster in FPGA rather than GPU [46]. Since FPGA has low power consumption, low latency, configurability, and parallelism, the combination of FPGA and CPU is utilized in multiple edge computing systems such as 5G networks [47], and real-time systems such as autonomous driving [48], video surveillance [49]. Regarding real-time services in edge computing, a proper task scheduler is needed to satisfy the real-time performance requirement of edge computing. The foundation of a task scheduler algorithm for real-time services is to assemble an appropriate graph of tasks, a directed acyclic graph (DAG), consisting of serial and parallel tasks. However, scheduling tasks based on the DAG representing a real-time application is an NP-hard problem [50].

Moreover, the storage capacity of edge computing is geographically distributed among edge servers. Furthermore, each edge server possesses its data communication capacity, which varies for each edge server.

4.5. 6G Networks

The sixth-generation mobile system (6G) standard and network goals are to enhance connectivity and service coverage, provide a reliable platform for vertical applications, and guarantee a reliable response time. The 6G is needed to fulfill the QoS and industrial control of Industry 4.0, the required data rate of extended reality (XR), the requirements of diverse AI applications, and to provide a reliable service for autonomous vehicles. The 6G deployment requirements are 1Tbps data rate, 1Kbps/Hz spectral efficiency, and latency of approximately μ-seconds; these specifications can be referred to as 6G TKμ. In [51], a novel task-centric three-layer decentralized model architecture is proposed for 6G, incorporating a super edge node. This architecture is designed to support various services and applications. In addition, a comprehensive review of reinforcement learning-based techniques for edge computing in 6G is presented in [52].

Moreover, combining 6G and edge computing provides a suitable platform for implementing federated learning techniques on edge devices for AI applications. However, the heterogeneity of edge devices and their limited resources pose challenges in the training process of federated learning and increase the training time. Therefore, the authors of [53] proposed a new approach to accelerate the federated learning technique.

5. Optimization Properties

Each study presents a formulation of the scheduling problem for desired applications in edge computing based on different criteria related to the application structure, environmental factors, and desired outcomes. The optimization properties can be classified based on the main viewpoint of the study, the number and type of factors considered for optimization, and the formulation method of the optimization problem [54].

5.1. Main Viewpoint

The optimization goals of studies on task scheduling in edge computing can be broadly categorized into three areas:

- End-user devices: the scheduling techniques consider optimizing parameters such as energy consumption, response time, or cost on the side of end-user devices.
- Edge servers: given the limited computational and storage resources of end-user devices, the proposed scheduling techniques aim to enhance the efficiency of edge servers. Specifically, these techniques strive to minimize energy consumption, improve resource utilization, and minimize costs.
- Hybrid: subsequent studies in this field have focused on hybrid scheduling techniques that optimize the parameters of both end-user devices and edge servers. These studies acknowledge that end-user devices have modest computational resources and explore the offloading of specific tasks to edge servers to improve the overall performance of the edge computing platform, including both end-user devices and edge servers.

5.2. Optimization Objective

Research papers on task scheduling can be classified as either single-objective or multi-objective. Single-objective papers attempt to optimize a single parameter, while multi-objective papers optimize multiple parameters simultaneously.

Each study set a goal for optimizing an aspect of the platform, such as response time, energy consumption, and cost.

According to the optimization goals and parameters, the optimization problem can be formulated by different methods, including Integer linear programming (ILP), mixed integer linear programming (MILP), mixed integer non-linear programming (MINLP), and Markov decision process (MDP).

6. RQ1: Centralized and Distributed Task Scheduling Techniques

Cloud computing is a centralized computation platform, while edge computing is a distributed one. In addition, edge servers possess limited resources compared to cloud servers. Finally, edge computing needs efficient scheduling techniques as it is dynamic, distributed, and heterogenous.

The scheduler has prior knowledge of future tasks in the offline scheduling algorithm. Therefore, the optimal offline scheduling algorithm can achieve the upper bound performance. On the contrary, there has yet to be prior knowledge about tasks in online scheduling algorithms. As soon as a task arrives in the system, the online scheduling algorithm realizes the task's parameters [55]. Although some of the offline scheduling algorithms mentioned in [29] outperform the online scheduling algorithms, the latest online scheduling algorithm performs better than offline scheduling algorithms. The advantage of online scheduling algorithms is attention to the dynamic of the edge computing network. One of the most significant steps in the offloading task to edge computing is allocating an appropriate resource to a task by considering the characteristics of both the resource and task. Therefore, task scheduling is a multi-objective optimization problem, which means jointly allocating communication, computing, and storage resources in edge computing is a multi-objective optimization problem [23]. Considerable hardships in resource scheduling for edge computing are the heterogeneity of computation platforms (CPU, GPU, FPGA) and resource limitations rather than cloud computing. Therefore, providing an appropriate task graph is an NP-hard problem. Several objectives, such as QoS, average latency, suitable edge server, and the number of edge users, must be considered [56]. According to the controlling manner, task scheduling techniques in edge computing are divided into centralized and distributed methods.

6.1. Centralized Task Scheduling Technique

Centralized methods offer a holistic system perspective, allowing them to make optimal decisions based on complete information about available resources and required tasks. Moreover, centralized methods are easier to manage and monitor because they permit centralized control of the system and centralized data storage. One of the centralized methods is convex optimization which involves finding the optimal solution to a

mathematical optimization problem subject to constraints. Approximation algorithms are another method that provides a near-optimal solution to a problem within a reasonable time frame. On the contrary, heuristic algorithms use a rule-based approach to find a solution that may not be optimal but is acceptable. Meta-heuristic algorithms, such as genetic algorithms and simulated annealing, are more complex approaches often used to find near-optimal solutions to complex problems. Finally, machine learning methods are increasingly being used for task scheduling in edge computing, with techniques such as reinforcement learning and deep learning showing promise in improving the efficiency and accuracy of task scheduling.

6.1.1. Convex Optimization

Task scheduling in edge computing is an NP-hard and "non-convex" problem; thus, the researchers attempted to convert the "non-convex" problem to a "near-convex" or "convex" optimization problem. Hence, the researchers can solve the scheduling problem with the convex optimization method as an offline task scheduling method. Lyapunov [57–60], decomposition [61], and alternating direction method of multipliers (ADMM) [62,63] techniques are mathematical optimization techniques utilized to solve "near-convex" or "convex" optimization problems. The proposed scheme in [63] aims to maximize the overall system reliability by intelligently allocating tasks to edge servers and cloud servers. The scheme consists of two main components: a task allocation algorithm and a task offloading algorithm. The task allocation algorithm allocates tasks to edge and cloud servers based on their reliability levels, computational capacity, and communication bandwidth. The task offloading algorithm determines the optimal offloading strategy for each task based on the reliability of the edge servers and cloud servers and the communication cost [63]. The proposed mechanism in [62], POTAM, considers computation and communication costs to allocate tasks to the most appropriate edge servers. The POTAM utilizes ADMM and a parallel algorithm to solve the large-scale task scheduling problem in a reasonable amount of time, which involves minimizing the total delay of executing multiple tasks on distributed edge servers while satisfying resource constraints [62]. The convex problem is solved with the Lyapunov technique in [64] in polynomial time, as the objective of task scheduling was minimizing the response time.

6.1.2. Approximation Algorithms

It is an offline task scheduling method, and many studies utilize an approximation algorithm to solve non-convex and NP-hard resource scheduling problems. The optimization problem is solved based on different approximation algorithms, including deterministic approximation algorithm [65] and local search-based approximation algorithm [66], or by utilizing different techniques such as MDP [67], K-means clustering, and hybrid quadratic programming [68] to solve an approximate method.

6.1.3. Heuristic

They are challenging to be a dynamic method for task scheduling due to the time-consuming and complex computation. In [69], a heuristic algorithm is proposed for task allocation and scheduling in the edge layer. This algorithm adopts a priority-based approach to allocate and schedule tasks in the edge layer based on their deadlines and computational requirements [69]. The study [70] presented a multi-objective optimization model for scheduling tasks in an integrated edge-cloud environment. The model includes makespan, energy consumption, and processing cost as the objectives. A heuristic algorithm is proposed to solve the model efficiently. Moreover, The scheduling algorithm utilized in [71] is a modified version of the earliest deadline first (EDF) algorithm, where tasks are prioritized based on their deadlines and processing time requirements at the edge servers. In addition, a two-step approach to solve the task scheduling problem is designed in [72]. In the first step, the algorithm determines the optimal task-to-device assignment that minimizes the overall completion time of the tasks, subject to the bandwidth con-

straints and the resource availability of the edge computing devices. In the second step, the algorithm schedules the tasks on the selected devices based on EDF [72]. Yi-Han et al. [73] proposed a task offloading and scheduling technique that considers dependent tasks in edge computing. Consequently, a joint cotask-aware offloading and scheduling (JCAOS) algorithm is presented that packs dependent tasks in "cotasks" and then creates a graph. In the next step, the algorithm schedules and offloads the "cotasks" regarding three parameters such as dependent degree, computational requirements, and network conditions. The scheduling algorithm prioritizes the task with the higher dependency. Since the JCAOS algorithm is a heuristic algorithm based on a first come first serve (FCFS) mechanism, it is hard for the JCAOS algorithm to provide a real-time solution for changes that happen in edge computing. The greedy algorithm is one of the heuristic methods utilized to solve resource scheduling in edge computing [70,74]. For example, the article [71] aggregates vehicular resources and proposes a latency-aware real-time scheduling framework (LARS) for offloading applications to appropriate vehicular resources in real time. The proposed framework includes a clustering-based algorithm for generating end-users and a greedy-based task-scheduling algorithm for offloading jobs to minimize job latency and maximize resource utilization [71].

6.1.4. Metaheuristic

They are proposed by utilizing different algorithms such as genetic algorithm [75–77], non-dominated sorting genetic algorithm (NSGA) [78–82], particle swarm optimization (PSO) [83], tabu search [84], ant colony [85,86], evolutionary algorithm [87], a combination of genetic algorithm and particle swarm optimization techniques [88]. In addition, the proposed technique in [89] employs the multi-objective whale optimization (MOWO) Algorithm, a meta-heuristic algorithm, to schedule tasks among multiple edge servers. The MOWO algorithm minimizes the system's overall energy consumption and total processing time while maximizing the number of completed tasks [89]. The paper [76] focused on the service placement problem concerning workload distribution across multiple edge servers for the applications. This problem is particularly relevant for IoT applications, as the limitations of edge computing resources and other objectives can lead to service level agreement (SLA) violations. To address this challenge, the authors of [76] propose a multi-objective genetic algorithm that combines random and heuristic solutions to generate near-optimal solutions. The designed algorithm in [76] aims to minimize SLA violations while considering competing load distribution and placement objectives. Furthermore, the VECMAN framework [90] includes a genetic algorithm for task scheduling while consisting of a heuristic algorithm for vehicle clustering and a machine-learning model for energy prediction.

6.1.5. Machine Learning

ML approaches utilizing deep learning [91] and reinforcement learning can achieve an optimal solution and are more compatible with the dynamic environment of edge computing rather than traditional static methods such as convex optimization and approximation [92]. MDP is suitable for modeling the online task scheduling problem in the dynamic environment of edge computing, and the reinforcement learning technique solves the model. Task scheduling in edge computing is defined as an optimization problem using MDP in [93]. The study proposes solving this optimization problem with deep reinforcement learning (DRL) to maximize the total reward in the dynamic environment of edge computing. However, the proposed DRL-based method in [93] only considers computational resources, disregarding memory, storage, and network bandwidth resources.In addition, the article [94] designed an online task scheduling framework for the edge computing platform to minimize task latency by optimizing offloading decisions, transmission power, and resource allocation. To this end, a DRL-based approach is developed, including a related and regularized stacked autoencoder that compresses data, an adaptive simulated annealing approach (ASA) for action search, and a preserved and prioritized experience replay mechanism (2pER) for training the policy network [94].

Moreover, one of the efficient types of reinforcement learning algorithms is Q-learning which is a value-based learning and an iterative algorithm learning the optimal Q-values [95]. Therefore, the paper [95] proposed an optimization approach where the goal is to minimize the execution time and power consumption of computational tasks by determining the optimal order and edge server for task offloading. Deep Q-learning network (DQN), as one of the DRL algorithms, addresses the dynamic environment of edge and efficiently solves the MDP model of online offloading problem [96]. A task allocation in a dynamic edge computing environment can be accomplished by combining DRLs shared experience replay mechanism with a DQN, as mentioned in [97]. Therefore, agents can learn from each other's experiences; each agent's experience is a batch of experiences stored in the replay buffer instead of only the most recent experiences of each agent. Accordingly, a centralized actor network (CAN) receives agents' policies and jointly trains them in a centralized manner, then CAN update each agent's Q-values. DQN demonstrates considerable results for online offloading problems in the dynamic environment of the edge [98–102], and the combination of DQN with long short-term memory (LSTM) [103] and federated learning (FL) [104] assemble appropriate schedulers for edge computing. In [26], Wang et al. proposed to solve the problem of online task scheduling with the meta-RL-based method, in which the second-order gradient in model-agnostic meta-learning (MAML) is replaced with a first-order approximation to reduce the cost of training.

6.2. Distributed Task Scheduling Techniques

Distributed methods are more fault-tolerant than centralized methods, as they can continue to operate even if some servers fail or leave the system. Since distributed methods allow the system to be easily expanded by adding new servers without requiring a significant overhaul of the system architecture, distributed methods are more scalable than centralized methods. Moreover, distributed methods can be more resilient to network disruptions or communication delays, as they allow for local decision-making without requiring constant communication with a central node. Furthermore, distributed methods enhance privacy and security by enabling local data processing on edge servers rather than sending them to a central location. Game theory, matching theory, auction, and FL are all examples of such techniques, each with unique strengths and applications. Game theory and matching theory are concerned with modeling the behavior of multiple decision-makers and finding optimal outcomes in complex situations. Auctions provide a mechanism for allocating resources or tasks among multiple bidders based on their bids. FL enables multiple parties to collaborate on training a machine learning model while preserving their data privacy. These techniques offer various benefits, including efficiency, fairness, privacy-preserving, and trust, making them applicable for task scheduling in the edge computing platform.

6.2.1. Game Theory

Game theory can schedule tasks in the edge computing [105–108]. Entities interact based on self-interest without intense complexity, and game theory analyzes the interactions. Smys et al. [109] proposed a task scheduling method based on game theory, which utilizes both cooperative and non-cooperative models. The method consists of three elements: (1) players, (2) their strategies in different situations, and (3) rewards. The objective of the game theory-based scheduler in [109] is to minimize waiting time; thus, to achieve this, the scheduler orders the tasks based on their deadlines.

In [110], they increased the task scheduling speed by utilizing the metadata of tasks in the scheduling procedure instead of receiving all data of tasks. Therefore, the communication overhead is significantly reduced. The goal of their scheduler is to maximize the efficiency of the edge system. Teng et al. [110] converted the scheduling problem to a cooperative game.

6.2.2. Matching Theory

It is a promising method for solving mixed-integer and the non-linear problem of task scheduling and resource management [111,112]. There is mutual matching between users and servers based on users' and servers' preferences; thus, there will be an association mapping between users and edge servers in task scheduling using match theory. Each edge server sorts tasks based on its preference relation in matching theory. The match graph (M) is a sub-graph of graph (I) that (M) has neither a common vertex nor an adjacent edge. In contrast, graph (I) is an initial graph of the whole edge computing network composed of all edge nodes, including edge servers and edge users. An energy-aware scheduler for edge computing is proposed by [113] using match theory by considering computation, communication, and delay limitations. In addition, low computational complexity is the objective of the scheduler in [114], and the scheduler considers both computation offloading and resource allocation.

6.2.3. Auction

Auction operates as the users publish their tasks and the rewards to the edge computing system. Then, edge servers analyze the rewards they can obtain through executing the tasks and submit their bids to the system. Finally, the system assigns the task to the edge node, which submitted the highest bids [115–118]; thus, the scheduler solves the task scheduling problem with a polynomial complexity that is near optimal. Several studies by [119] utilized the auction method to provide a solution for the task scheduling problem.

6.2.4. Distributed Machine Learning

An algorithm named P2D3PG, based on distributed deep reinforcement learning, is proposed in [120]. This algorithm aims to maximize cache hit rates on edge devices while preserving users' data. Since edge caching represents a distributed optimization problem, the authors of [120] formulate it as a model-free Markov decision process. By employing an appropriate caching method, the algorithm reduces bandwidth overhead and facilitates the implementation of computation-intensive and time-sensitive applications.

Moreover, edge caching plays a crucial role in the Industrial Internet of Things (IIoT) as it enables real-time control and application. However, accurately predicting popularity patterns among devices in edge computing requires significant time and effort. The challenges associated with predicting popularity include limited data in data-sensitive systems and the high cost of labeling in supervised learning. As a result, a secure unsupervised framework is proposed in [121] for predicting the popularity of IIoT in edge computing. The popularity prediction problem is formulated using a model-free Markov chain in [121]. The authors of [121] address this problem by dividing popularity into local and global measures, and they propose an unsupervised recurrent federated learning algorithm.

FL is a distributed ML algorithm developed by Google researchers; FL distributes the DRL algorithm, and FL trains DRL agents in a distributed manner. In addition, since edge computing is a distributed computing paradigm, FL as a distributed ML is a suitable technique to solve the resource scheduling problem [122–126]. The traditional ML techniques are centralized and utilize cloud infrastructure for storage and computation demands. However, the idea of FL is to implement DRL algorithms in a distributed manner and it is possible to deploy ML services near the users via edge computing; thus, FL can mitigate all communication costs, data privacy, and legalization concerns. FL is a collaborative learning method suitable for geographically distributed edge servers to train task scheduling algorithms with online responses to the dynamic behavior of edge computing networks and without intense data communication and private data relocation. FL is studied in time-critical industrial applications with a massive quantity of sensitive data. FL enables Industrial Internet of Things (IIoT) devices to train and develop an intelligent framework for task scheduling [127]. Since estimating the exact execution time is hard, especially in the Internet of Vehicles (IoV), providing an optimal task scheduling algorithm is a substantial challenge. Therefore, a task scheduling technique is proposed by [128] based on

FL by considering the power-delay product parameter for optimization. A context-aware scheduling algorithm is proposed by [129] to schedule VR tasks over an edge network, and FL implements DRL on edge nodes. According to [129], FL provides the proper solutions for training with a distributed dataset, updating the scheduler algorithms with available nodes in unstable communication.

Finally, the advantages and disadvantages of both centralized and distributed task scheduling techniques are provided in Table 5, and a comprehensive comparison of the different criteria is provided in Table 6.

Table 5. Comparison of advantages and disadvantages of current task scheduling techniques.

Technique	Operation Manner	Advantages	Disadvantages
Convex optimization	Centralized	(1) Mature and widely used. (2) The sub-optimal optimization results are Easily achievable.	(1) Complex and time-consuming calculations. (2) Challenging to implement in systems.
Approximation	Centralized	(1) Flexible and straightforward to be implemented. (2) A local search algorithm can be simply designed for the most difficult NP-hard problems.	(1) Local optimum issue. (2) Unreliable performance due to inherent randomness of approximation manner.
Heuristic methods	Centralized	(1) Mature and widely used. (2) Efficient.	(1) Complex and time-consuming calculations. (2) Challenging to be compatible with the dynamic environment of edge computing. (3) Local optimal issue.
Meta-heuristic methods	Centralized	(1) Mature and widely used. (2) Efficient.	(1) Many parameters should be defined in Meta-heuristic methods. (2) Hard to adjust the parameters of meta-heuristic algorithm.
Machine Learning	Centralized	(1) Strong parallel processing capability. (2) Strong distributed storage and learning capability.	(1) Require a large number of parameters. (2) A black-box process and learning process cannot be observed. (3) Long learning time. (4) Local optimal issue.
Game Theory	Distributed	(1) Straightforward to implement.	(1) Mutual solution may not be the optimal solution. (2) Requirement of Nash Equilibrium and its continuous calculation.
Matching Theory	Distributed	(1) Suitable for high dynamic network. (2) Practical for complex network.	(1) Suitable for binary offloading instead of partially offloading.
Federated Learning	Distributed	(1) The training process is distributed between nodes; thus, uploading data to a server is unnecessary. (2) As data are not uploaded, it is beneficial for user privacy. (3) Low transmission rate. (4) Low training time.	(1) Numerous devices are involved. (2) Vulnerable to malicious communication attacks.

Table 6. Comprehensive comparison of selected papers on task scheduling in edge computing.

Reference	Main Viewpoint	Optimization Goal	Objective Number	Modeling Problem	Utilized Technique	Applicable for Real-Time Task Scheduling
[57]	edge servers	Energy	Single	MINLP	Convex optimization (Lyapunov technique)	No
[58]	End-user devices	Privacy, Energy	Multiple	ILP	Convex optimization (Lyapunov technique)	No
[59]	edge servers	Energy	Single	MINLP	Convex optimization (Lyapunov technique)	No
[60]	edge servers	Time, Energy, Data transmission	Multiple	MINLP	Convex optimization (Lyapunov technique)	No
[90]	End-user devices	Energy	Single	MILP	Meta-heuristic (genetic algorithm)	No
[69]	End-user devices	Time	Single	MILP	Heuristic (EDF)	Yes
[130]	End-user devices	QoE	Multiple	MDP	Machine learning (deep reinforcement learning)	No
[77]	End-user devices	Time	Single	MIP	Meta-heuristic (genetic algorithm)	Yes
[131]	End-user devices	Energy, Time, Cost	Multiple	MDP	Machine learning (deep reinforcement learning)	No
[89]	edge servers	Energy	Multiple	MILP	Meta-heuristic (Whale Optimization Algorithm)	No
[75]	edge servers	Energy, Time	Multiple	ILP	Meta-heuristic (genetic algorithm)	Yes
[132]	End-user devices	Energy, Time	Multiple	ILP	Machine learning (deep reinforcement learning)	Yes
[70]	edge servers	Energy	Single	MINLP	Heuristic (semi-greedy)	Yes
[92]	Hybrid	Energy, Time	Multiple	MILP	Machine learning (deep learning)	No
[91]	End-user devices	Energy	Single	MINLP	Machine learning (deep learning)	Yes
[94]	End-user devices	Time	Single	MINLP	Machine learning (deep reinforcement learning)	No
[71]	End-user devices	Time	Single	MILP	Heuristic (Greedy Algorithm)	Yes
[76]	Hybrid	Time, Cost	Multiple	MINLP	Meta-heuristic (genetic algorithm)	No
[133]	End-user devices	Energy, Time	Multiple	MDP	Machine learning (deep reinforcement learning)	No
[134]	Hybrid	Time, Energy	Multiple	MDP	Machine learning (deep reinforcement learning)	No
[135]	Hybrid	Energy, QoS	Multiple	MDP	Machine learning (deep reinforcement learning)	No
[136]	End-user devices	Energy, Task finish ratio	Multiple	MDP	Machine learning (deep reinforcement learning)	No
[61]	End-user devices	Time	Single	MINLP	Heuristic	No
[63]	edge server	Time	Single	MILP	Heuristic	No
[62]	edge server	Time, Cost	Multiple	MILP	Heuristic	No
[64]	edge server	Energy, Time	Multiple	ILP	Heuristic (Lyapunov)	Yes
[65]	edge server	Cost	Single	MILP	Approximation	No
[73]	Hybrid	Energy	Single	MILP	Heuristic (variation of FCFS)	No
[74]	edge server	Time, Energy	Multiple	MILP	Heuristic (Greedy Algorithm)	No
[87]	edge server	Time, Energy, Cost	Multiple	MINLP	Meta-heuristic (Evolutionary Algorithm)	No
[76]	Hybrid	Service Level Agreement	Multiple	MINLP	Meta-heuristic (Genetic Algorithm)	No

Table 6. *Cont.*

Reference	Main Viewpoint	Optimization Goal	Objective Number	Modeling Problem	Utilized Technique	Applicable for Real-Time Task Scheduling
[85]	End-user devices	Time	Single	ILP	Meta-heuristic (Ant colony)	No
[86]	End-user devices	Energy	Single	MILP	Meta-heuristic (Ant colony)	No
[88]	End-user devices	Energy	Single	MINLP	Meta-heuristic (Genetic algorithm + Particle swarm optimization)	No
[78]	End-user devices	Time, Energy	Multiple	MILP	Meta-heuristic (NSGA-III)	No
[72]	End-user devices	Time	Single	ILP	Heuristic (EDF)	Yes
[59]	edge servers	Energy	Single	MINLP	Convex optimization (Lyapunov technique)	No
[60]	edge servers	Time, Energy, Data transmission	Multiple	MINLP	Convex optimization (Lyapunov technique)	No
[90]	End-user devices	Energy	Single	MILP	Meta-heuristic (genetic algorithm)	No
[69]	End-user devices	Time	Single	MILP	Heuristic (EDF)	Yes
[130]	End-user devices	QoE	Multiple	MDP	Machine learning (deep reinforcement learning)	No
[77]	End-user devices	Time	Single	MIP	Meta-heuristic (genetic algorithm)	Yes
[131]	End-user devices	Energy, Time, Cost	Multiple	MDP	Machine learning (deep reinforcement learning)	No
[89]	edge servers	Energy	Multiple	MILP	Meta-heuristic (Whale Optimization Algorithm)	No
[75]	edge servers	Energy, Time	Multiple	ILP	Meta-heuristic (genetic algorithm)	Yes
[132]	End-user devices	Energy, Time	Multiple	ILP	Machine learning (deep reinforcement learning)	Yes
[70]	edge servers	Energy	Single	MINLP	Heuristic (semi-greedy)	Yes
[92]	Hybrid	Energy, Time	Multiple	MILP	Machine learning (deep learning)	No
[91]	End-user devices	Energy	Single	MINLP	Machine learning (deep learning)	Yes
[94]	End-user devices	Time	Single	MINLP	Machine learning (deep reinforcement learning)	No
[71]	End-user devices	Time	Single	MILP	Heuristic (Greedy Algorithm)	Yes
[76]	Hybrid	Time, Cost	Multiple	MINLP	Meta-heuristic (genetic algorithm)	No
[85]	End-user devices	Time	Single	ILP	Meta-heuristic (Ant colony)	No
[86]	End-user devices	Energy	Single	MILP	Meta-heuristic (Ant colony)	No
[88]	End-user devices	Energy	Single	MINLP	Meta-heuristic (Genetic algorithm + Particle swarm optimization)	No
[78]	End-user devices	Time, Energy	Multiple	MILP	Meta-heuristic (NSGA-III)	No
[72]	End-user devices	Time	Single	ILP	Heuristic (EDF)	Yes

7. RQ2: Scheduling Real-Time Embedded System Application Tasks

Table 5 presents ten studies [64,69–72,75,77,91,94,132] that are suitable for scheduling real-time systems on edge computing. The majority of these studies use heuristic approaches to schedule tasks [64,69–72], while the remaining studies employ distinct methods. Although heuristic techniques are mature enough, they can suffer from delays due to their complex and time-consuming computations. Therefore, utilizing the heuristic method in a highly dynamic edge computing environment can be challenging. In addition to timing issues, the local optima problem is common in heuristic techniques for scheduling tasks in edge computing environments, which have many variables and parameters.

The meta-heuristic methods, particularly genetic algorithms [75,77], are utilized for task scheduling in edge computing based on Table 5. The meta-heuristic approaches

encounter challenges such as high computational complexity, lengthy calculation time to achieve acceptable accuracy, and slow convergence during the optimization process and period of evolution [91]. Therefore, meta-heuristic algorithms may not be suitable for real-time system optimization in dynamic environments. Furthermore, the efficiency of meta-heuristic-based approaches decreases as the number of users on the edge computing platform increases. In contrast, machine learning methods offer faster adaptability, decision-making, and scalability, providing significant advantages over heuristic algorithms and meta-heuristic techniques [132].

According to Table 5, some studies employ centralized machine learning approaches such as deep learning [91] and deep reinforcement learning [94,132]. While machine learning techniques increase computation speed through parallel processing, the learning time of ML can become lengthy when considering a large number of parameters in an edge computing environment. As a result, centralized ML techniques might not be efficient for scheduling time-sensitive edge computing applications.

However, distributed ML techniques can rectify the long run-time of the training procedure by increasing parallel processing and decreasing data communication overhead in edge computing [137]. Accordingly, there is no need to gather all the data on a single machine when the data are inherently distributed over the network. FL is a distributed ML implementation that enables the exploitation of distributed resources. In FL, all users collaborate in the training phase without moving their data to a server. In addition to facilitating the training phase, FL increases the security and privacy of users' data by eliminating the need for users to move their data to an external location. Therefore, FL is a promising technique for systems with sensitive data, including healthcare and financial applications.

8. Challenges and Future Research Directions

Despite significant progress in research on resource management and task scheduling in edge computing, several important issues regarding integrating real-time systems with edge computing still need to be fully explored. This section discusses the current challenges and suggests potential research directions for future research.

8.1. Requirements of Realtime Systems

Real-time systems have strict QoS requirements, including response time, throughput, and reliability. Real-time tasks require the availability of various resources, including computation, storage, data, input/output, and others, prior to their deadlines. Therefore, timely task execution within the deadline is critical for real-time systems. Failure to meet deadlines can result in a decrease in QoS and consequences of varying severity, depending on whether the system is a soft or hard real-time system [138]. Ensuring tasks' deadlines in real-time applications employing scheduling techniques is already challenging in a single-processor system. However, the complexity increases in edge computing, which is a distributed system with heterogeneous computational and storage resources spread across the network. Alongside computational and storage features, communication plays a crucial role that must be considered. Therefore, the scheduling technique needs to consider the various communication bandwidths to ensure low latency in data transmission while scheduling tasks for real-time applications. Most existing task-scheduling techniques in edge computing only consider computational resources when scheduling tasks. However, a practical task-scheduling technique needs to consider communication and storage parameters, in addition to computing parameters, to provide a realistic solution for scheduling tasks in edge computing. Future studies should focus on proposing reliable task-scheduling techniques for the edge computing platform that fully meet the real-time system requirements. These techniques should consider the computational, communication, and storage resources of edge computing. Evaluation of these techniques can be based on their miss-ratio to demonstrate their effectiveness in task scheduling on the edge computing platform.

8.2. Dynamic Environments and Tasks Dependancy

Since devices frequently move, join, and leave the network, the edge computing environment is dynamic. As a result, the set of tasks that need to be executed would change; thus, a proper scheduling algorithm is needed to adapt to these changes and maintain acceptable effectiveness quickly. The scheduling algorithm should schedule the new tasks and previous tasks together in a way that none of the tasks of real-time application miss their deadline. Consequently, the task scheduling technique should respond fast enough to the change in the set of tasks in edge computing. However, the mobility characteristics of devices are often overlooked in most existing works, even though user mobility was one of the driving factors behind the creation of edge computing platforms. Consequently, the task scheduler must track the trajectory of devices and consider them when scheduling tasks. In addition, the tasks depend on each other in the real world, while to the best of our knowledge, most existing works only consider the independent task model instead of the dependent one [138]. Therefore, future work on task scheduling in edge computing should consider task dependencies in their scheduling procedure. Consequently, developing a holistic task scheduling algorithm is necessary to handle task dependencies and adapt to network changes effectively.

8.3. Security and Privacy

The real-time systems implemented by edge devices often process sensitive data and typically serve a monitoring and controlling role, such as surveillance cameras, fire and smoke alarms. Comparatively, the conventional security techniques are too much to be executed on the edge users; thus, lightweight security techniques or frameworks are needed to be developed specifically for edge computing [139]. In addition, conventional trust management algorithms are excessively demanding for edge devices. Therefore, the development of trust management algorithms compatible with the limited resources of edge devices is crucial. This area holds great potential as a promising research topic for future exploration. Furthermore, conventional cryptographic techniques are known to be resource intensive. Consequently, implementing these techniques on edge devices is impracticable; thus, there is a pressing need for edge-compatible cryptographic techniques [140]. Furthermore, considering that edge computing comprises various software and hardware components, it is critical to establish unified security schemes that can adequately support and address the inherent heterogeneity of the platform.

Finally, the security-aware scheduling algorithms can prevent data breaches during processing, ensure that the return result from edge servers is not corrupted, and authorize both the edge server and user in the edge computing platform. Additionally, since the edge computing architecture spans multiple layers, it is vulnerable to hostile attacks. Consequently, fault-tolerant task scheduling algorithms are needed to handle edge server failures.

9. Conclusions

Edge computing is a distributed computing architecture composed of geographically distributed edge servers with heterogeneous processing units such as CPUs, GPUs, DSPs, and FPGAs. Therefore, edge computing is a suitable platform for IoT devices to offload computational tasks and utilize the storage capacity of edge servers. This review aims to review the state-of-the-art studies on task scheduling comprehensively approaches in edge computing. The review discusses task scheduling techniques, which can be centralized or distributed. Centralized techniques require a server to compute, store, monitor, and control the scheduler's decisions. The centralized techniques include convex optimization, approximation algorithms, heuristics, meta-heuristics, and machine learning techniques. Each method has specific advantages and disadvantages that make it suitable for different applications and environments. Alternatively, distributed techniques, such as game theory, matching theory, auction, and federated learning, are designed to execute over multiple devices, eliminating the need for a single server. By leveraging the power of distributed computing, these techniques accelerate computation by parallel processing and offer a

more scalable and fault-tolerant solution for task scheduling. Consequently, distributed techniques reduce the data communication overhead and keep user data safe and secure by not moving it to an external server. The advantages and disadvantages of centralized and distributed techniques are clarified.

Task scheduling is an optimization problem that involves setting specific goals and employing a method to formulate the problem. State-of-the-art papers on task scheduling employed different approaches, including integer linear programming (ILP), mixed integer linear programming (MILP), mixed integer non-linear programming (MINLP), and Markov decision process (MDP) to formulate the optimization problem. After conducting a thorough comparison, meta-heuristic and machine learning approaches are the most promising techniques for scheduling tasks in real-time edge computing applications. These techniques are capable of handling the dynamic environment of edge computing effectively.

The challenges facing task scheduling in edge computing for time-sensitive applications are managing the QoS requirement, handling dynamic environments, and addressing security and privacy concerns. Each of these challenges represents a potential research direction for future investigations.

Author Contributions: Writing—original draft preparation, A.A. (Amin Avan); supervision and writing—review and editing, A.A. (Akramul Azim); supervision and writing—review and editing, Q.H.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

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

References

1. Ali, B.; Gregory, M.A.; Li, S. Multi-access edge computing architecture, data security and privacy: A review. *IEEE Access* **2021**, *9*, 18706–18721. [CrossRef]
2. Islam, A.; Debnath, A.; Ghose, M.; Chakraborty, S. A survey on task offloading in multi-access edge computing. *J. Syst. Archit.* **2021**, *118*, 102225. [CrossRef]
3. Ahmed, A.A.; Echi, M. Hawk-eye: An ai-powered threat detector for intelligent surveillance cameras. *IEEE Access* **2021**, *9*, 63283–63293. [CrossRef]
4. Gupta, B. Analysis of IoT concept applications: Smart home perspective. In Proceedings of the Future Access Enablers for Ubiquitous and Intelligent Infrastructures: 5th EAI International Conference, FABULOUS 2021, Virtual Event, 6–7 May 2021; Volume 382, p. 167.
5. Rana, B.; Singh, Y.; Singh, P.K. A systematic survey on internet of things: Energy efficiency and interoperability perspective. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4166. [CrossRef]
6. Atieh, A.T. The next generation cloud technologies: A review on distributed cloud, fog and edge computing and their opportunities and challenges. *Res. Rev. Sci. Technol.* **2021**, *1*, 1–15.
7. Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Niyato, D.; Dobre, O.; Poor, H.V. 6G Internet of Things: A comprehensive survey. *IEEE Internet Things J.* **2021**, *9*, 359–383. [CrossRef]
8. Deng, Y.; Chen, X.; Zhu, G.; Fang, Y.; Chen, Z.; Deng, X. Actions at the Edge: Jointly Optimizing the Resources in Multi-access Edge Computing. *IEEE Wirel. Commun.* **2022**, *29*, 192–198. [CrossRef]
9. Busacca, F.; Galluccio, L.; Palazzo, S. Drone-assisted edge computing: A game-theoretical approach. In Proceedings of the IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 6–9 July 2020; pp. 671–676.
10. Shannigrahi, S.; Mastorakis, S.; Ortega, F.R. Next-generation networking and edge computing for mixed reality real-time interactive systems. In Proceedings of the 2020 IEEE International Conference on Communications Workshops (ICC Workshops), Dublin, Ireland, 7–11 June 2020; pp. 1–6.
11. Cui, M.; Zhong, S.; Li, B.; Chen, X.; Huang, K. Offloading autonomous driving services via edge computing. *IEEE Internet Things J.* **2020**, *7*, 10535–10547. [CrossRef]
12. Qiu, T.; Chi, J.; Zhou, X.; Ning, Z.; Atiquzzaman, M.; Wu, D.O. Edge computing in industrial internet of things: Architecture, advances and challenges. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2462–2488. [CrossRef]
13. Chen, Y.-Y.; Lin, Y.-H.; Hu, Y.-C.; Hsia, C.-H.; Lian, Y.-A.; Jhong, S.-Y. Distributed Real-Time Object Detection Based on Edge-Cloud Collaboration for Smart Video Surveillance Applications. *IEEE Access* **2022**, *10*, 93745–93759. [CrossRef]
14. Hartmann, M.; Hashmi, U.S.; Imran, A. Edge computing in smart health care systems: Review, challenges, and research directions. *Trans. Emerg. Telecommun. Technol.* **2022**, *33*, e3710. [CrossRef]

15. Sacco, A.; Esposito, F.; Marchetto, G. Restoring Application Traffic of Latency-Sensitive Networked Systems Using Adversarial Autoencoders. *IEEE Trans. Netw. Serv. Manag.* **2022**, *19*, 2521–2535. [CrossRef]
16. Satyanarayanan, M.; Bahl, P.; Caceres, R.; Davies, N. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Comput.* **2009**, *8*, 14–23. [CrossRef]
17. Ray, P.P.; Dash, D.; De, D. Edge computing for Internet of Things: A survey, e-healthcare case study and future direction. *J. Netw. Comput. Appl.* **2019**, *140*, 1–22. [CrossRef]
18. Cao, K.; Hu, S.; Shi, Y.; Colombo, A.W.; Karnouskos, S.; Li, X. A survey on edge and edge-cloud computing assisted cyber-physical systems. *IEEE Trans. Ind. Inform.* **2021**, *17*, 7806–7819. [CrossRef]
19. Mansouri, Y.; Babar, M.A. A review of edge computing: Features and resource virtualization. *J. Parallel Distrib. Comput.* **2021**, *150*, 155–183. [CrossRef]
20. Carvalho, G.; Cabral, B.; Pereira, V.; Bernardino, J. Edge computing: Current trends, research challenges and future directions. *Computing* **2021**, *103*, 993–1023. [CrossRef]
21. Shakarami, A.; Ghobaei-Arani, M.; Masdari, M.; Hosseinzadeh, M. A survey on the computation offloading approaches in mobile edge/cloud computing environment: A stochastic-based perspective. *J. Grid Comput.* **2020**, *18*, 639–671. [CrossRef]
22. Liu, L.; Chen, C.; Pei, Q.; Maharjan, S.; Zhang, Y. Vehicular edge computing and networking: A survey. *Mob. Netw. Appl.* **2021**, *26*, 1145–1168. [CrossRef]
23. Luo, Q.; Hu, S.; Li, C.; Li, G.; Shi, W. Resource scheduling in edge computing: A survey. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 2131–2165. [CrossRef]
24. Chen, S.; Li, Q.; Zhou, M.; Abusorrah, A. Recent advances in collaborative scheduling of computing tasks in an edge computing paradigm. *Sensors* **2021**, *21*, 779. [CrossRef] [PubMed]
25. Mijuskovic, A.; Chiumento, A.; Bemthuis, R.; Aldea, A.; Havinga, P. Resource management techniques for cloud/fog and edge computing: An evaluation framework and classification. *Sensors* **2021**, *21*, 1832. [CrossRef] [PubMed]
26. Salaht, F.A.; Desprez, F.; Lebre, A. An overview of service placement problem in fog and edge computing. *ACM Comput. Surv. CSUR* **2020**, *53*, 1–35. [CrossRef]
27. Jiang, C.; Fan, T.; Gao, H.; Shi, W.; Liu, L.; Cérin, C.; Wan, J. Energy aware edge computing: A survey. *Comput. Commun.* **2020**, *151*, 556–580. [CrossRef]
28. Lin, H.; Zeadally, S.; Chen, Z.; Labiod, H.; Wang, L. A survey on computation offloading modeling for edge computing. *J. Netw. Comput. Appl.* **2020**, *169*, 102781. [CrossRef]
29. 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]
30. Jia, M.; Fan, Y.; Cai, Y. A Survey on Task Scheduling Schemes in Mobile Edge Computing. In Proceedings of the Big Data and Security: Third International Conference, ICBDS 2021, Shenzhen, China, 26–28 November 2021; Proceedings. Springer: Berlin/Heidelberg, Germany, 2022; pp. 426–439.
31. Avan, A.; Taheri, M.; Moaiyeri, M.H.; Navi, K. Energy-Efficient approximate compressor design for error-resilient digital signal processing. *Int. J. Electron.* **2022**, 1–23. [CrossRef]
32. Avan, A.; Maleknejad, M.; Navi, K. High-speed energy efficient process, voltage and temperature tolerant hybrid multi-threshold 4: 2 compressor design in CNFET technology. *IET Circuits Devices Syst.* **2020**, *14*, 357–368. [CrossRef]
33. Busacca, F.; Grasso, C.; Palazzo, S.; Schembra, G. A smart road side unit in a microeolic box to provide edge computing for vehicular applications. *IEEE Trans. Green Commun. Netw.* **2022**, *7*, 194–210. [CrossRef]
34. Hayat, S.; Jung, R.; Hellwagner, H.; Bettstetter, C.; Emini, D.; Schnieders, D. Edge computing in 5G for drone navigation: What to offload? *IEEE Robot. Autom. Lett.* **2021**, *6*, 2571–2578. [CrossRef]
35. Zhang, G.; Ni, S.; Zhao, P. Learning-Based Joint Optimization of Energy Delay and Privacy in Multiple-User Edge-Cloud Collaboration MEC Systems. *IEEE Internet Things J.* **2021**, *9*, 1491–1502. [CrossRef]
36. Zhang, H.; Yang, Y.; Shang, B.; Zhang, P. Joint Resource Allocation and Multi-Part Collaborative Task Offloading in MEC Systems. *IEEE Trans. Veh. Technol.* **2022**, *71*, 8877–8890. [CrossRef]
37. Choi, J.D.; Kim, M.Y. A sensor fusion system with thermal infrared camera and LiDAR for autonomous vehicles and deep learning based object detection. *ICT Express* **2022**, *9*, 222–227. [CrossRef]
38. Liang, S.; Wu, H.; Zhen, L.; Hua, Q.; Garg, S.; Kaddoum, G.; Hassan, M.M.; Yu, K. Edge YOLO: Real-time intelligent object detection system based on edge-cloud cooperation in autonomous vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 25345–25360. [CrossRef]
39. Gao, Y.; Yang, S.; Li, F.; Trajanovski, S.; Zhou, P.; Hui, P.; Fu, X. Video Content Placement At the Network Edge: Centralized and Distributed Algorithms. *IEEE Trans. Mob. Comput.* **2022**, 1–17. [CrossRef]
40. Kitchenham, B.; Brereton, O.P.; Budgen, D.; Turner, M.; Bailey, J.; Linkman, S. Systematic literature reviews in software engineering—A systematic literature review. *Inf. Softw. Technol.* **2009**, *51*, 7–15. [CrossRef]
41. Mahjoubi, A.; Grinnemo, K.-J.; Taheri, J. EHGA: A Genetic Algorithm Based Approach for Scheduling Tasks on Distributed Edge-Cloud Infrastructures. In Proceedings of the 2022 13th International Conference on Network of the Future (NoF), Ghent, Belgium, 5–7 October 2022; pp. 1–5.

42. Mahjoubi, A.; Taheri, J.; Grinnemo, K.-J.; Deng, S. Optimal Placement of Recurrent Service Chains on Distributed Edge-Cloud Infrastructures. In Proceedings of the 2021 IEEE 46th Conference on Local Computer Networks (LCN), Edmonton, AB, Canada, 4–7 October 2021; pp. 495–502.
43. Fizza, K.; Auluck, N.; Azim, A. Improving the schedulability of real-time tasks using fog computing. *IEEE Trans. Serv. Comput.* **2019**, *15*, 372–385. [CrossRef]
44. Mahjoubi, A.; Grinnemo, K.-J.; Taheri, J. An Efficient Simulated Annealing-based Task Scheduling Technique for Task Offloading in a Mobile Edge Architecture. In Proceedings of the 2022 IEEE 11th International Conference on Cloud Networking (CloudNet), Paris, France, 7–10 November 2022; pp. 159–167.
45. Zhu, Z.; Zhang, J.; Zhao, J.; Cao, J.; Zhao, D.; Jia, G.; Meng, Q. A hardware and software task-scheduling framework based on CPU+ FPGA heterogeneous architecture in edge computing. *IEEE Access* **2019**, *7*, 148975–148988. [CrossRef]
46. Boutros, A.; Nurvitadhi, E.; Ma, R.; Gribok, S.; Zhao, Z.; Hoe, J.C.; Betz, V.; Langhammer, M. Beyond peak performance: Comparing the real performance of AI-optimized FPGAs and GPUs. In Proceedings of the 2020 International Conference on Field-Programmable Technology (ICFPT), Maui, HI, USA, 9–11 December 2020; pp. 10–19.
47. Yan, L.; Cao, S.; Gong, Y.; Han, H.; Wei, J.; Zhao, Y.; Yang, S. SatEC: A 5G satellite edge computing framework based on microservice architecture. *Sensors* **2019**, *19*, 831. [CrossRef]
48. Du, S.; Huang, T.; Hou, J.; Song, S.; Song, Y. FPGA based acceleration of game theory algorithm in edge computing for autonomous driving. *J. Syst. Archit.* **2019**, *93*, 33–39. [CrossRef]
49. Cho, G.; Kim, S.-H.; Youn, C.-H. Hybrid Resource Scheduling Scheme for Video Surveillance in GPU-FPGA Accelerated Edge Computing System. In *Advances in Artificial Intelligence and Applied Cognitive Computing*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 679–694.
50. Simon, B. Scheduling Task Graphs on Modern Computing Platforms. Ph.D. Thesis, Université de Lyon, Lyon, France, 2018.
51. You, X.; Huang, Y.; Liu, S.; Wang, D.; Ma, J.; Xu, W.; Zhang, C.; Zhan, H.; Zhang, C.; Zhang, J. Toward 6G TK $\mu $ Extreme Connectivity: Architecture, Key Technologies and Experiments. *arXiv* **2022**, arXiv:2208.01190.
52. Wei, P.; Guo, K.; Li, Y.; Wang, J.; Feng, W.; Jin, S.; Ge, N.; Liang, Y.-C. Reinforcement learning-empowered mobile edge computing for 6G edge intelligence. *IEEE Access* **2022**, *10*, 65156–65192. [CrossRef]
53. He, J.; Guo, S.; Li, M.; Zhu, Y. AceFL: Federated Learning Accelerating in 6G-enabled Mobile Edge Computing Networks. *IEEE Trans. Netw. Sci. Eng.* **2022**, *10*, 1364–1375. [CrossRef]
54. Goudarzi, M. Energy and Time Aware Scheduling of Applications in Edge and Fog Computing Environments. 2022. Available online: https://www.researchgate.net/publication/361431337_Energy_and_Time_Aware_Scheduling_of_Applications_in_Edge_and_Fog_Computing_Environments (accessed on 8 May 2023).
55. Atoui, W.S.; Ajib, W.; Boukadoum, M. Offline and online scheduling algorithms for energy harvesting RSUs in VANETs. *IEEE Trans. Veh. Technol.* **2018**, *67*, 6370–6382. [CrossRef]
56. Liu, H.; Long, X.; Li, Z.; Long, S.; Rong, R.; Wang, H.-M. Joint Optimization of Request Assignment and Computing Resource Allocation in Multi-Access Edge Computing. *IEEE Trans. Serv. Comput.* **2022**, *16*, 1254–1267. [CrossRef]
57. Chen, L.; Zhou, S.; Xu, J. Computation peer offloading for energy-constrained mobile edge computing in small-cell networks. *IEEE/ACM Trans. Netw.* **2018**, *26*, 1619–1632. [CrossRef]
58. He, X.; Jin, R.; Dai, H. Peace: Privacy-preserving and cost-efficient task offloading for mobile-edge computing. *IEEE Trans. Wirel. Commun.* **2019**, *19*, 1814–1824. [CrossRef]
59. Zhang, Q.; Gui, L.; Hou, F.; Chen, J.; Zhu, S.; Tian, F. Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN. *IEEE Internet Things J.* **2020**, *7*, 3282–3299. [CrossRef]
60. Li, C.; Tang, J.; Luo, Y. Dynamic multi-user computation offloading for wireless powered mobile edge computing. *J. Netw. Comput. Appl.* **2019**, *131*, 1–15. [CrossRef]
61. Saleem, U.; Liu, Y.; Jangsher, S.; Tao, X.; Li, Y. Latency minimization for D2D-enabled partial computation offloading in mobile edge computing. *IEEE Trans. Veh. Technol.* **2020**, *69*, 4472–4486. [CrossRef]
62. Zhong, X.; Wang, X.; Yang, T.; Yang, Y.; Qin, Y.; Ma, X. POTAM: A parallel optimal task allocation mechanism for large-scale delay sensitive mobile edge computing. *IEEE Trans. Commun.* **2022**, *70*, 2499–2517. [CrossRef]
63. Liang, J.; Ma, B.; Feng, Z.; Huang, J. Reliability-aware Task Processing and Offloading for Data-intensive Applications in Edge computing. *IEEE Trans. Netw. Serv. Manag.* **2023**, 1. [CrossRef]
64. Deng, Y.; Chen, Z.; Yao, X.; Hassan, S.; Ibrahim, A.M. Parallel offloading in green and sustainable mobile edge computing for delay-constrained IoT system. *IEEE Trans. Veh. Technol.* **2019**, *68*, 12202–12214. [CrossRef]
65. Pasteris, S.; Wang, S.; Herbster, M.; He, T. Service placement with provable guarantees in heterogeneous edge computing systems. In Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications, Paris, France, 29 April–2 May 2019; pp. 514–522.
66. Lu, S.; Wu, J.; Duan, Y.; Wang, N.; Fang, J. Cost-efficient resource provision for multiple mobile users in fog computing. In Proceedings of the 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), Tianjin, China, 4–6 December 2019; pp. 422–429.
67. Meng, X.; Wang, W.; Wang, Y.; Lau, V.K.; Zhang, Z. Closed-form delay-optimal computation offloading in mobile edge computing systems. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 4653–4667. [CrossRef]

68. Guo, Y.; Wang, S.; Zhou, A.; Xu, J.; Yuan, J.; Hsu, C.H. User allocation-aware edge cloud placement in mobile edge computing. *Softw. Pract. Exp.* **2020**, *50*, 489–502. [CrossRef]
69. Stavrinides, G.L.; Karatza, H.D. A hybrid approach to scheduling real-time IoT workflows in fog and cloud environments. *Multimed. Tools Appl.* **2019**, *78*, 24639–24655. [CrossRef]
70. Azizi, S.; Shojafar, M.; Abawajy, J.; Buyya, R. Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach. *J. Netw. Comput. Appl.* **2022**, *201*, 103333. [CrossRef]
71. Hu, S.; Li, G.; Shi, W. Lars: A latency-aware and real-time scheduling framework for edge-enabled internet of vehicles. *IEEE Trans. Serv. Comput.* **2021**, *16*, 398–411. [CrossRef]
72. Meng, J.; Tan, H.; Xu, C.; Cao, W.; Liu, L.; Li, B. Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing. In Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications, Paris, France, 29 April–2 May 2019; pp. 2287–2295.
73. Chiang, Y.-H.; Zhang, T.; Ji, Y. Joint cotask-aware offloading and scheduling in mobile edge computing systems. *IEEE Access* **2019**, *7*, 105008–105018. [CrossRef]
74. Ben Salah, N.; Bellamine Ben Saoud, N. An IoT-oriented Multiple Data Replicas Placement Strategy in Hybrid Fog-Cloud Environment. In Proceedings of the 2021 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, Virtual Event, 31 May–2 June 2021; pp. 119–128.
75. Hoseiny, F.; Azizi, S.; Shojafar, M.; Ahmadiazar, F.; Tafazolli, R. PGA: A priority-aware genetic algorithm for task scheduling in heterogeneous fog-cloud computing. In Proceedings of the IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Vancouver, BC, Canada, 10–13 May 2021; pp. 1–6.
76. Maia, A.M.; Ghamri-Doudane, Y.; Vieira, D.; de Castro, M.F. An improved multi-objective genetic algorithm with heuristic initialization for service placement and load distribution in edge computing. *Comput. Netw.* **2021**, *194*, 108146. [CrossRef]
77. Aburukba, R.O.; 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]
78. Xu, X.; Liu, Q.; Luo, Y.; Peng, K.; Zhang, X.; Meng, S.; Qi, L. A computation offloading method over big data for IoT-enabled cloud-edge computing. *Future Gener. Comput. Syst.* **2019**, *95*, 522–533. [CrossRef]
79. Peng, K.; Zhu, M.; Zhang, Y.; Liu, L.; Zhang, J.; Leung, V.; Zheng, L. An energy-and cost-aware computation offloading method for workflow applications in mobile edge computing. *EURASIP J. Wirel. Commun. Netw.* **2019**, *2019*, 1–15. [CrossRef]
80. Xu, X.; Cao, H.; Geng, Q.; Liu, X.; Dai, F.; Wang, C. Dynamic resource provisioning for workflow scheduling under uncertainty in edge computing environment. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e5674. [CrossRef]
81. Hu, S.; Li, G. Dynamic request scheduling optimization in mobile edge computing for IoT applications. *IEEE Internet Things J.* **2019**, *7*, 1426–1437. [CrossRef]
82. Xu, X.; Li, Y.; Huang, T.; Xue, Y.; Peng, K.; Qi, L.; Dou, W. An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks. *J. Netw. Comput. Appl.* **2019**, *133*, 75–85. [CrossRef]
83. Mseddi, A.; Jaafar, W.; Elbiaze, H.; Ajib, W. Joint container placement and task provisioning in dynamic fog computing. *IEEE Internet Things J.* **2019**, *6*, 10028–10040. [CrossRef]
84. Wu, Y.; Wu, J.; Chen, L.; Yan, J.; Luo, Y. Efficient task scheduling for servers with dynamic states in vehicular edge computing. *Comput. Commun.* **2020**, *150*, 245–253. [CrossRef]
85. Kishor, A.; Chakarbarty, C. Task offloading in fog computing for using smart ant colony optimization. *Wirel. Pers. Commun.* **2021**, *127*, 1683–1704. [CrossRef]
86. Huang, P.-Q.; Wang, Y.; Wang, K.; Liu, Z.-Z. A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing. *IEEE Trans. Cybern.* **2019**, *50*, 4228–4241. [CrossRef]
87. Hussain, M.; Azar, A.T.; Ahmed, R.; Umar Amin, S.; Qureshi, B.; Dinesh Reddy, V.; Alam, I.; Khan, Z.I. SONG: A Multi-Objective Evolutionary Algorithm for Delay and Energy Aware Facility Location in Vehicular Fog Networks. *Sensors* **2023**, *23*, 667. [CrossRef]
88. Guo, F.; Zhang, H.; Ji, H.; Li, X.; Leung, V.C. An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing. *IEEE/ACM Trans. Netw.* **2018**, *26*, 2651–2664. [CrossRef]
89. Chen, W.; Wang, D.; Li, K. Multi-user multi-task computation offloading in green mobile edge cloud computing. *IEEE Trans. Serv. Comput.* **2018**, *12*, 726–738. [CrossRef]
90. Bahreini, T.; Brocanelli, M.; Grosu, D. VECMAN: A framework for energy-aware resource management in vehicular edge computing systems. *IEEE Trans. Mob. Comput.* **2021**, *22*, 1231–1245. [CrossRef]
91. Jiang, F.; Wang, K.; Dong, L.; Pan, C.; Xu, W.; Yang, K. Deep-learning-based joint resource scheduling algorithms for hybrid MEC networks. *IEEE Internet Things J.* **2019**, *7*, 6252–6265. [CrossRef]
92. Huang, L.; Feng, X.; Feng, A.; Huang, Y.; Qian, L.P. Distributed deep learning-based offloading for mobile edge computing networks. *Mob. Netw. Appl.* **2018**, *27*, 1123–1130. [CrossRef]
93. Sheng, S.; Chen, P.; Chen, Z.; Wu, L.; Yao, Y. Deep reinforcement learning-based task scheduling in iot edge computing. *Sensors* **2021**, *21*, 1666. [CrossRef]
94. Jiang, F.; Wang, K.; Dong, L.; Pan, C.; Yang, K. Stacked autoencoder-based deep reinforcement learning for online resource scheduling in large-scale MEC networks. *IEEE Internet Things J.* **2020**, *7*, 9278–9290. [CrossRef]
95. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]

96. Qiu, X.; Liu, L.; Chen, W.; Hong, Z.; Zheng, Z. Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing. *IEEE Trans. Veh. Technol.* **2019**, *68*, 8050–8062. [CrossRef]
97. Nguyen, T.T.; Nguyen, N.D.; Nahavandi, S. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Trans. Cybern.* **2020**, *50*, 3826–3839. [CrossRef] [PubMed]
98. Liu, X.; Yu, J.; Wang, J.; Gao, Y. Resource allocation with edge computing in IoT networks via machine learning. *IEEE Internet Things J.* **2020**, *7*, 3415–3426. [CrossRef]
99. Wang, J.; Hu, J.; Min, G.; Zhan, W.; Ni, Q.; Georgalas, N. Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning. *IEEE Commun. Mag.* **2019**, *57*, 64–69. [CrossRef]
100. Zhang, K.; Zhu, Y.; Leng, S.; He, Y.; Maharjan, S.; Zhang, Y. Deep learning empowered task offloading for mobile edge computing in urban informatics. *IEEE Internet Things J.* **2019**, *6*, 7635–7647. [CrossRef]
101. Xiong, X.; Zheng, K.; Lei, L.; Hou, L. Resource allocation based on deep reinforcement learning in IoT edge computing. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 1133–1146. [CrossRef]
102. Zhai, Y.; Bao, T.; Zhu, L.; Shen, M.; Du, X.; Guizani, M. Toward reinforcement-learning-based service deployment of 5G mobile edge computing with request-aware scheduling. *IEEE Wirel. Commun.* **2020**, *27*, 84–91. [CrossRef]
103. Lu, H.; Gu, C.; Luo, F.; Ding, W.; Liu, X. Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning. *Future Gener. Comput. Syst.* **2020**, *102*, 847–861. [CrossRef]
104. Shen, S.; Han, Y.; Wang, X.; Wang, Y. Computation offloading with multiple agents in edge-computing–supported IoT. *ACM Trans. Sens. Netw. TOSN* **2019**, *16*, 1–27. [CrossRef]
105. Li, Q.; Zhao, J.; Gong, Y. Cooperative computation offloading and resource allocation for mobile edge computing. In Proceedings of the 2019 IEEE International Conference on Communications Workshops (ICC Workshops), Shanghai, China, 20–24 May 2019; pp. 1–6.
106. Ranadheera, S.; Maghsudi, S.; Hossain, E. Computation offloading and activation of mobile edge computing servers: A minority game. *IEEE Wirel. Commun. Lett.* **2018**, *7*, 688–691. [CrossRef]
107. Zhang, J.; Xia, W.; Yan, F.; Shen, L. Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing. *IEEE Access* **2018**, *6*, 19324–19337. [CrossRef]
108. Asheralieva, A.; Niyato, D. Hierarchical game-theoretic and reinforcement learning framework for computational offloading in UAV-enabled mobile edge computing networks with multiple service providers. *IEEE Internet Things J.* **2019**, *6*, 8753–8769. [CrossRef]
109. Smys, S.; Ranganathan, G. Performance evaluation of game theory based efficient task scheduling for edge computing. *J. ISMAC* **2020**, *2*, 50–61.
110. Teng, H.; Li, Z.; Cao, K.; Long, S.; Guo, S.; Liu, A. Game theoretical task offloading for profit maximization in mobile edge computing. *IEEE Trans. Mob. Comput.* **2022**, 1. [CrossRef]
111. Gu, B.; Zhou, Z. Task offloading in vehicular mobile edge computing: A matching-theoretic framework. *IEEE Veh. Technol. Mag.* **2019**, *14*, 100–106. [CrossRef]
112. Chiti, F.; Fantacci, R.; Paganelli, F.; Picano, B. Virtual functions placement with time constraints in fog computing: A matching theory perspective. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 980–989. [CrossRef]
113. Gu, B.; Zhou, Z.; Mumtaz, S.; Frascolla, V.; Bashir, A.K. Context-aware task offloading for multi-access edge computing: Matching with externalities. In Proceedings of the 2018 IEEE global communications conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6.
114. Pham, Q.-V.; Leanh, T.; Tran, N.H.; Park, B.J.; Hong, C.S. Decentralized computation offloading and resource allocation for mobile-edge computing: A matching game approach. *IEEE Access* **2018**, *6*, 75868–75885. [CrossRef]
115. Zhang, D.; Tan, L.; Ren, J.; Awad, M.K.; Zhang, S.; Zhang, Y.; Wan, P.-J. Near-optimal and truthful online auction for computation offloading in green edge-computing systems. *IEEE Trans. Mob. Comput.* **2019**, *19*, 880–893. [CrossRef]
116. Ma, L.; Wang, X.; Wang, X.; Wang, L.; Shi, Y.; Huang, M. TCDA: Truthful combinatorial double auctions for mobile edge computing in industrial Internet of Things. *IEEE Trans. Mob. Comput.* **2021**, *21*, 4125–4138. [CrossRef]
117. Peng, X.; Ota, K.; Dong, M. Multiattribute-based double auction toward resource allocation in vehicular fog computing. *IEEE Internet Things J.* **2020**, *7*, 3094–3103. [CrossRef]
118. Zhou, H.; Wu, T.; Chen, X.; He, S.; Guo, D.; Wu, J. Reverse auction-based computation offloading and resource allocation in mobile cloud-edge computing. *IEEE Trans. Mob. Comput.* **2022**, 1–15. [CrossRef]
119. He, J.; Zhang, D.; Zhou, Y.; Zhang, Y. A truthful online mechanism for collaborative computation offloading in mobile edge computing. *IEEE Trans. Ind. Inform.* **2019**, *16*, 4832–4841. [CrossRef]
120. Liu, S.; Zheng, C.; Huang, Y.; Quek, T.Q. Distributed reinforcement learning for privacy-preserving dynamic edge caching. *IEEE J. Sel. Areas Commun.* **2022**, *40*, 749–760. [CrossRef]
121. Zheng, C.; Liu, S.; Huang, Y.; Zhang, W.; Yang, L. Unsupervised Recurrent Federated Learning for Edge Popularity Prediction in Privacy-Preserving Mobile-Edge Computing Networks. *IEEE Internet Things J.* **2022**, *9*, 24328–24345. [CrossRef]
122. Nguyen, D.C.; Ding, M.; Pham, Q.-V.; Pathirana, P.N.; Le, L.B.; Seneviratne, A.; Li, J.; Niyato, D.; Poor, H.V. Federated learning meets blockchain in edge computing: Opportunities and challenges. *IEEE Internet Things J.* **2021**, *8*, 12806–12825. [CrossRef]
123. Wang, R.; Lai, J.; Zhang, Z.; Li, X.; Vijayakumar, P.; Karuppiah, M. Privacy-preserving federated learning for internet of medical things under edge computing. *IEEE J. Biomed. Health Inform.* **2022**, *27*, 854–865. [CrossRef]

124. Lakhan, A.; Mohammed, M.A.; Kadry, S.; AlQahtani, S.A.; Maashi, M.S.; Abdulkareem, K.H. Federated learning-aware multi-objective modeling and blockchain-enable system for IIoT applications. *Comput. Electr. Eng.* **2022**, *100*, 107839. [CrossRef]
125. Shi, T.; Tian, H.; Zhang, T.; Loo, J.; Ou, J.; Fan, C.; Yang, D. Task Scheduling with Collaborative Computing of MEC System Based on Federated Learning. In Proceedings of the 2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring), Helsinki, Finland, 19–22 June 2022; pp. 1–5.
126. Zhang, Y.; Zhang, X.; Cai, Y. Multi-task Federated Learning based on Client Scheduling in Mobile Edge Computing. In Proceedings of the 2022 IEEE/CIC International Conference on Communications in China (ICCC), Foshan, China, 11–13 August 2022; pp. 185–190.
127. Zhang, L.; Wu, S.; Xu, H.; Liu, Q.; Hong, C.S.; Han, Z. Federated Learning Over the Industrial Internet of Things: A Joint Optimization of Edge Association and Resource Allocation. 2022. Available online: https://www.techrxiv.org/articles/preprint/Federated_Learning_Over_the_Industrial_Internet_of_Things_A_Joint_Optimization_of_Edge_Association_and_Resource_Allocation/20784001 (accessed on 8 May 2023).
128. Sun, F.; Zhang, Z.; Zeadally, S.; Han, G.; Tong, S. Edge Computing-Enabled Internet of Vehicles: Towards Federated Learning Empowered Scheduling. *IEEE Trans. Veh. Technol.* **2022**, *71*, 10088–10103. [CrossRef]
129. Shahidinejad, A.; Farahbakhsh, F.; Ghobaei-Arani, M.; Malik, M.H.; Anwar, T. Context-aware multi-user offloading in mobile edge computing: A federated learning-based approach. *J. Grid Comput.* **2021**, *19*, 1–23. [CrossRef]
130. Lu, H.; He, X.; Du, M.; Ruan, X.; Sun, Y.; Wang, K. Edge QoE: Computation offloading with deep reinforcement learning for Internet of Things. *IEEE Internet Things J.* **2020**, *7*, 9255–9265. [CrossRef]
131. Wang, J.; Hu, J.; Min, G.; Zhan, W.; Zomaya, A.Y.; Georgalas, N. Dependent task offloading for edge computing based on deep reinforcement learning. *IEEE Trans. Comput.* **2021**, *71*, 2449–2461. [CrossRef]
132. Yeganeh, S.; Sangar, A.B.; Azizi, S. A novel Q-learning-based hybrid algorithm for the optimal offloading and scheduling in mobile edge computing environments. *J. Netw. Comput. Appl.* **2023**, *214*, 103617. [CrossRef]
133. Cheng, Z.; Min, M.; Liwang, M.; Huang, L.; Gao, Z. Multiagent DDPG-based joint task partitioning and power control in Fog computing networks. *IEEE Internet Things J.* **2021**, *9*, 104–116. [CrossRef]
134. Cheng, Z.; Liwang, M.; Chen, N.; Huang, L.; Du, X.; Guizani, M. Deep reinforcement learning-based joint task and energy offloading in UAV-aided 6G intelligent edge networks. *Comput. Commun.* **2022**, *192*, 234–244. [CrossRef]
135. Zhou, X.; Liang, W.; Yan, K.; Li, W.; Kevin, I.; Wang, K.; Ma, J.; Jin, Q. Edge-Enabled Two-Stage Scheduling Based on Deep Reinforcement Learning for Internet of Everything. *IEEE Internet Things J.* **2022**, *10*, 3295–3304. [CrossRef]
136. Chen, Y.; Gu, W.; Li, K. Dynamic task offloading for internet of things in mobile edge computing via deep reinforcement learning. *Int. J. Commun. Syst.* **2022**, e5154. [CrossRef]
137. Verbraeken, J.; Wolting, M.; Katzy, J.; Kloppenburg, J.; Verbelen, T.; Rellermeyer, J.S. A survey on distributed machine learning. *Acm Comput. Surv. Csur* **2020**, *53*, 1–33. [CrossRef]
138. Kopetz, H.; Steiner, W. *Real-Time Systems: Design Principles for Distributed Embedded Applications*; Springer Nature: Berlin/Heidelberg, Germany, 2022.
139. Alwarafy, A.; Al-Thelaya, K.A.; Abdallah, M.; Schneider, J.; Hamdi, M. A survey on security and privacy issues in edge-computing-assisted internet of things. *IEEE Internet Things J.* **2020**, *8*, 4004–4022. [CrossRef]
140. Yahuza, M.; Idris, M.Y.I.B.; Wahab, A.W.B.A.; Ho, A.T.; Khan, S.; Musa, S.N.B.; Taha, A.Z.B. Systematic review on security and privacy requirements in edge computing: State of the art and future research opportunities. *IEEE Access* **2020**, *8*, 76541–76567. [CrossRef]