Resource Scheduling in Edge Computing: A Survey

Quyuan Luo*, Shihong Hu*, Changle Li*, Senior Member, IEEE, Guanghui Li*, and Weisong Shi*, Fellow, IEEE

Abstract—With the proliferation of the Internet of Things (IoT) and the wide penetration of wireless networks, the burgeoning demand for data communications and computing calls for the emerging edge computing paradigm. By moving the services and functions located in the cloud to the proximity of users, edge computing can provide powerful communication, storage, networking, and communication capacity. The resource scheduling in edge computing, which is the key to the success of edge computing systems, has attracted increasing research interests. In this paper, we survey the state-of-the-art research findings to know the research progress in this field. Specifically, we present the architecture of edge computing, under which different collaborative manners for resource scheduling are discussed. Particularly, we introduce a unified model before summarizing the current works on resource scheduling from three research issues, including computation offloading, resource allocation, and resource provisioning. Based on two modes of operation, i.e., centralized and distributed modes, different techniques for resource scheduling are discussed and compared. Also, we summarize the main performance indicators based on the surveyed literature. To shed light on the significance of resource scheduling in real-world scenarios, we discuss several typical application scenarios involved in the research of resource scheduling in edge computing. Finally, we highlight some open research challenges yet to be addressed and outline several open issues as the future research direction.

I. INTRODUCTION

A. From Cloud Computing to Edge Computing

WITH the rapid development of the mobile Internet, smart devices have become an indispensable part of people’s life. Increasingly complex applications such as mobile payment, smart healthcare, mobile games, and virtual reality (VR) put higher requirements on the resource capacity of smart devices. Since Google put forward the concept of cloud computing in 2008 [1], cloud computing was gradually accepted and introduced into the mobile environment, which breaks through the resource limitations of smart devices and provides highly demanding applications for users. Cloud computing is a cost-effective model that provides abundant applications and services while making information technology (IT) management more accessible and responding to users’ demands faster [2]. The services (computing, communication, storage, and all necessary services) are delivered and implemented in a simplified way: on-demand, regardless of the users’ location and the type of smart devices.

Thanks to rapid advances in underlying technologies, the Internet of Things (IoT) is opening tremendous opportunities for a large number of novel applications that promise to improve the quality of our lives [3]. Technically, all applications we discussed in this survey belong to the category of IoT. Applications such as unmanned aerial vehicle (UAV), connected and autonomous vehicle (CAV), video service, smart city, smart health, smart manufactury, and smart home are all committed to improving the quality of our lives through various technologies of IoT. However, in recent years, the IoT era has brought higher requirements for transmission bandwidth, latency, energy consumption, application performance, and reliability. In this context, due to the limited bandwidth, high latency, and high energy consumption in the centralized processing model of cloud computing, it is hard to meet the high-performance requirements of users. Fortunately, it can be estimated that tens of billions of edge nodes (ENs) will be deployed in the near future [4]. By integrating these large amounts of idle resources distributed at the edge of the network to seamlessly provide services for users, a new computing paradigm - edge computing is proposed, which is regarded as the key technology and architectural concept of the transition to 5G [5]. Fig. 1 illustrates the edge computing...
computing (MEC), has been introduced. The MEC brings communication, a new emerging concept, known as mobile edge computing, to cope with the issue in mobile computing, it is hard to meet the high-performance requirements of users. In this context, due to the limited bandwidth, high latency, and quality-of-service (QoS) can be obtained by users. Let’s take the edge computing in mobile communication/5G communication moves the services and functions originally located in the cloud to the proximity of users, which integrates the cloud computing platform and the network to provide powerful computing, storage, networking, and communication capacity at the edge of the network. Edge computing is interchangeable with fog computing, but edge computing focuses more on the things side, while fog computing focuses more on the infrastructure side [6]. Since the services and functions are closer to users in edge computing, a better quality-of-experience (QoE) and quality-of-service (QoS) can be obtained by users. Let’s take the edge computing in mobile communication as an example. With the development of mobile communication, especially the 5G communication, the demand for high-quality wireless services shows a trend of exponential growth. In the age of 5G, in addition to mobile phones, tablets, a lot of new business scenarios in mobile network service emerges, such as autonomous driving, VR, and augmented reality (AR), and more close to the life business scenarios, such as smart grid, smart agriculture, smart city, and environmental monitoring. The emergence of these new service scenarios has higher requirements for 5G key technical indicators such as time delay, energy efficiency and reliability. In this context, due to the limited bandwidth, high latency, and high energy consumption in the centralized processing model of cloud computing, it is hard to meet the high-performance requirements of users. To cope with the issue in mobile communication, a new emerging concept, known as mobile edge computing (MEC), has been introduced. The MEC brings computation and storage resources to the edge of the mobile network enabling it to run the highly demanding applications at the user equipment while meeting strict performance requirements [7].

B. Resource Scheduling in Edge Computing

In recent years, resource scheduling in edge computing has attracted widespread interest from industry and academia. Before introducing resource scheduling in edge computing, two questions should be answered firstly:

1) What is Resource Scheduling in Edge Computing?

Generally, resource scheduling refers to the set of actions and methodology that participants used to efficiently assign resources to the tasks that need to complete, and achieve the objectives of participants based on resource availability. Specifically, according to edge computing characteristics, the key terms of resource scheduling in edge computing can be detailed as follows.

- **Resources**: Various resources existing in the edge network, by which the powerful serviceability is provided and the tasks can be completed. The resource in edge network can be categorized into three types, i.e., communication resources, storage resources (also as caching resources), and computing resources [8], [9].

- **Tasks**: Tasks generally refer to data generated from users. The task types may vary based on different application scenarios for different objectives. For example, the data from LiDAR and high-definition camera on CAVs is for safety purpose [10], [11]; the data from body area networks (BAN) is for health monitoring; and the data from surveillance cameras is for security [12].

- **Participants**: To complete tasks, there are different collaborative processing modes that involves different participants. For “things-edge collaboration”, users (referred as “things”) and edge are the participants [13]. For “things-edge-cloud collaboration”, users, edge, and cloud are the participants [14]. For “edge-cloud collaboration”, edge and cloud center are the participants [15].

- **Objectives**: Different users pursue different objectives during task processing. For example, CAVs aim to obtain low latency for traffic safety [16]. UAVs and smart health devices aim to reduce energy consumption for long battery life [17]. The objectives can also be referred to as performance indicators.

- **Actions**: The ways to achieve the objectives of participants are referred to as actions. In edge computing, there are mainly three actions: 1) computation offloading, which decides whether a task is offloaded to the edge or the cloud to process [18]; 2) resource allocation, which means allocating the communication, storage resources, and computing resources for tasks [19]; 3) resource provisioning, which decides the user-resource pair association from the perspective of users, or actively conducts resource placement from the perspective of service providers (SPs) [20], [21].
Methodology: Methodology refers to the methods, techniques, and algorithms to better take the above actions for the objectives of participants. Basically, the methodology can be mainly categorized into centralized and distributed manners. The centralized methodology needs a control center to collect global information while the distributed methodology does not [22], [23].

2) Why do We Need Resource Scheduling in Edge Computing? While edge computing greatly strengthens the serviceability of edge network by providing powerful computing, storage, and communication capacities, it also requires appropriate resource scheduling strategies from three perspectives.

User: Tens of billions of heterogeneous end-devices are geographically deployed in a distributed manner, the data volume generated from those end-devices and their corresponding applications are also heterogeneous. Orchestrating the limited edge resources to better process those data requires appropriate resource scheduling strategies. In the edge computing network, there are not only static end-devices (e.g., sensors in smart homes, video cameras in public places), but also dynamic ones such as UAVs and vehicles, making the resource management even more challenging. Appropriate resource scheduling can alleviate this situation. Besides, the data from different application scenarios may have different service requirements. For example, the CAVs in intelligent transportation systems (ITS) need to process data within several milliseconds for traffic safety; thus low latency is their main objective. The UAV-assisted edge computing usually focuses more on long battery life; thus the objective of low energy consumption is expected during data processing. Also, some mobile devices (MDs) and IoT devices aim to achieve low data processing cost. Therefore, it needs proper resource scheduling strategies to meet those service requirements.

Service provider: In addition to users, the edge computing ecosystem incorporates multiple actors, such as edge infrastructure SPs, edge computing service providers, application service providers, and mobile network operators. Although these SPs and operators are resource-rich and have powerful serviceability, they are all commercial entities aiming at earning revenue by providing services [35]. In this context, designing an appropriate resource scheduling strategy can help them get a maximal revenue during service providing competition at a minimal cost.

Edge network: Edge resources are distributed and scattered in the edge network. It is a waste of resources if scattered ones can not be efficiently utilized by resource scheduling. For example, the parked vehicles (PVs) account for a large portion of the global vehicles and have idle time to perform computational workloads [36], [37]. If an efficient resource strategy is applied, they can be combined to establish an available and cost-effective computing resource pool [38], which helps to alleviate workloads of edge computing servers and promote the distributed computing environment. Besides, since both users and SPs try to earn their benefits from edge computing, it is more like a game between buyers and sellers in terms of resources and services. An effective resource strategy can jointly consider their interests and improve the edge system utility [39].

C. Related Surveys

In recent years, many surveys on edge computing from various perspectives have been published, as shown in Table I. Mao et al. [24] presented a survey with the focus of joint radio-and-computational resource management in edge computing. Likely, a more recent survey [30] also focused on resource management in edge computing. The difference is that this survey is from the viewpoint of architecture, infrastructure, and the underlying algorithms about resource management. Furthermore, both [28] and [31] presented a comprehensive survey of resource management in edge computing, the work in [28] surveyed related literature in terms of resource type, objective, resource location, and resource while Ghobaei-Arani et al. [31] provided a systematic review from application placement, resource scheduling, task offloading, load balancing, resource allocation and provisioning six fields in resource management. Wang et al. [25] summarized the related works on computing, caching, and communication techniques in the area of edge computing. Mach and Becvar [7] surveyed the research on computation offloading in the area of edge computing. Later, Lin et al. [10] presented a more comprehensive survey on computation offloading. The review angle of the survey [26] is more macro. It comprehensively elaborated on the definition, architecture, application areas, and advantages of edge computing. Besides, Varghese et al. [34] presented a systematic survey on edge benchmarking, which summarized the research from the system under test, techniques, quality metrics, and benchmark runtime in the edge computing. Some surveys focus on one topic, like service adoption and provision [27], resource provision from a machine learning perspective [29] or computing paradigms [33] in edge computing.

It can be concluded that some existing surveys summarized the research in edge computing only from a single angle in the resource scheduling field, like computation offloading or resource provisioning. Some surveys in previous years mostly discussed topics in edge computing from a high level and failed to comprehensively address these topics at the depth. With the increasing enthusiasm of the academic community for edge computing research in recent years, a large number of new research results have emerged, among which the research on resource scheduling is particularly prominent. Although the existing surveys listed in Table I have reviewed edge computing from various perspectives, none of them focus on the resource scheduling issue in a comprehensive way. This motivates us to present a systematic survey on resource scheduling, so we review from multiple perspectives, including architecture, research issue, techniques, indicators, and applications to provide a comprehensive, informative and up-to-date viewpoint for researchers.
TABLE I
A SUMMARY OF SURVEYS ON EDGE COMPUTING

<table>
<thead>
<tr>
<th>Paper</th>
<th>Year</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wang et al. [25]</td>
<td>2017</td>
<td>Issues on computing, caching and communication techniques in edge computing.</td>
</tr>
<tr>
<td>Abbas et al. [26]</td>
<td>2017</td>
<td>Relevant research and technological developments in edge computing.</td>
</tr>
<tr>
<td>Peng et al. [27]</td>
<td>2018</td>
<td>Service adoption and provision in edge computing.</td>
</tr>
<tr>
<td>Toce et al. [28]</td>
<td>2018</td>
<td>Resource management and optimization of multiple resources in edge computing.</td>
</tr>
<tr>
<td>Duc et al. [29]</td>
<td>2019</td>
<td>Resource provisioning in Edge-Cloud computing from a machine learning perspective.</td>
</tr>
<tr>
<td>Hong et al. [30]</td>
<td>2019</td>
<td>Resource management from the architecture, infrastructure and algorithms in edge computing.</td>
</tr>
<tr>
<td>Ghobaei et al. [31]</td>
<td>2019</td>
<td>Resource management approaches in edge computing.</td>
</tr>
<tr>
<td>Santos et al. [32]</td>
<td>2019</td>
<td>Resource provisioning from theory to practice in edge computing</td>
</tr>
<tr>
<td>Ren et al. [33]</td>
<td>2019</td>
<td>Issues on different computing paradigms in edge computing.</td>
</tr>
<tr>
<td>Varghese et al. [34]</td>
<td>2020</td>
<td>Different dimensions of research works in edge benchmarking.</td>
</tr>
</tbody>
</table>

Fig. 2. The distribution of papers surveyed by year and source. Book includes books and book chapters; Report includes arXiv articles, website articles and white papers; Conference includes conference and symposium papers; Journal includes journal and magazine articles.

D. Contribution and Organization

This article provides a comprehensive survey of the state-of-the-art research with a focus on resource scheduling in edge computing. Fig. 2 shows the distribution of papers surveyed by year and source. Specifically, the focus of this article is five-fold.

- **Architecture (Section II):** A three-tier edge computing architecture including the thing layer, the edge layer, and the cloud layer is first introduced. Then we elaborate on four different collaborations for resource scheduling under the three-tier architecture, i.e., things-edge, things-edge-cloud, edge-edge, and edge-cloud.

- **Basic Model and Research issue (Section III):** To achieve the different requirements of both end-devices and the system for QoS and QoE, several basic models are first introduced. Based on those models, we then present three aspects involved in resource scheduling, which forms the three key research issues, i.e., computation offloading, resource allocation, and resource provisioning.

- **Technique and indicator (Section IV):** We summarize the main performance indicators such as latency, energy consumption, cost, utility, profit, and resource utilization in existing works. To achieve those objectives, we also elaborate on the resource scheduling techniques both in centralized and distributed ways.

- **Application (Section V):** We summarize several typical application scenarios involved in the research on resource scheduling in edge computing, mainly including UAV, CAV, video service, smart city, smart health, smart manufacturing, and smart home.

- **Challenge and open issue (Section VI):** The lessons learned in the area of resource scheduling in edge computing are highlighted and several challenges yet to be addressed are presented for future research directions.

To help the readers have a comprehensive picture of the structure of this survey, Fig. 3 outlines the organization of the survey, and Table II lists the acronyms that will be frequently used in the survey.

II. ARCHITECTURE

This section introduces the edge computing architecture for resource scheduling. We overview the composition of the architecture and introduce a three-tier heterogeneous edge computing network, where the first tier is the thing layer, the second tier is the edge layer, and the third one is the cloud layer. Based on the three-tier architecture, we then present different collaborative manners for resource scheduling in edge computing.

A. Overview of the Architecture for Resource Scheduling in Edge Computing

Traditional cloud computing has difficulty to meet the high requirements of users in real-time response and low energy consumption due to bandwidth congestion and heavy load on the core network (CN). Nevertheless, the edge computing paradigm itself cannot be a substitute for cloud computing because it does not have as powerful resource capacity as cloud computing. In some cases, however, the advantages of edge computing can be leveraged to offload computing services from the cloud to the edge to improve users’ QoE. Accordingly, cloud computing and edge computing are complementary and mutually reinforcing. Thus, the resource scheduling in edge computing is not only operated among users and the edge, but also among users, the edge, and the cloud. The three-tier heterogeneous architecture for resource scheduling in edge computing is presented, as shown in Fig. 4, including the thing layer (a.k.a, the user...
The three-tier architecture is a widely popular and accepted paradigm by many existing works [7], [10], [24], [26], [30]. The function of this kind of architecture is to illustrate the relationship among components that make up the edge computing system. In the following, we first give a brief introduction on the

---

**TABLE II**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADMM</td>
<td>Alternating Direction Method of Multipliers</td>
<td>MD</td>
<td>Mobile Device</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
<td>MDC</td>
<td>Micro Data Center</td>
</tr>
<tr>
<td>AR</td>
<td>Augmented Reality</td>
<td>MDP</td>
<td>Markov Decision Process</td>
</tr>
<tr>
<td>BAN</td>
<td>Body Area Network</td>
<td>MEC</td>
<td>Mobile Edge Computing</td>
</tr>
<tr>
<td>BS</td>
<td>Base Station</td>
<td>MILP</td>
<td>Mixed Integer Linear Programming</td>
</tr>
<tr>
<td>CAV</td>
<td>Connected and Autonomous Vehicle</td>
<td>MU</td>
<td>Mobile User</td>
</tr>
<tr>
<td>CC</td>
<td>Computing and Communication</td>
<td>NFV</td>
<td>Network Function Virtualization</td>
</tr>
<tr>
<td>CCS</td>
<td>Computing, Communication, and Storage</td>
<td>NSGA</td>
<td>Non-dominated Sorting Genetic Algorithm</td>
</tr>
<tr>
<td>CN</td>
<td>Core Network</td>
<td>NOMA</td>
<td>Non-orthogonal Multiple Access</td>
</tr>
<tr>
<td>DQN</td>
<td>Deep Q-network</td>
<td>PV/EC</td>
<td>Parked Vehicle Edge Computing</td>
</tr>
<tr>
<td>DRL</td>
<td>Deep Reinforcement Learning</td>
<td>PSI</td>
<td>Particle Swarm Optimization</td>
</tr>
<tr>
<td>DSRC</td>
<td>Dedicated Short-Range Communications</td>
<td>PV</td>
<td>Parked Vehicle</td>
</tr>
<tr>
<td>EC</td>
<td>Edge Cloud</td>
<td>QoE</td>
<td>Quality of Experience</td>
</tr>
<tr>
<td>EG</td>
<td>Edge Gateway</td>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>BN</td>
<td>Edge Node</td>
<td>RSU</td>
<td>Road Side Unit</td>
</tr>
<tr>
<td>ES</td>
<td>Edge Server</td>
<td>SP</td>
<td>Service Provider</td>
</tr>
<tr>
<td>FiWi</td>
<td>Fiber-Wireless</td>
<td>SCA</td>
<td>Successive Convex Approximation</td>
</tr>
<tr>
<td>FL</td>
<td>Federated Learning</td>
<td>SDN</td>
<td>Soft-defined Network</td>
</tr>
<tr>
<td>GA</td>
<td>Genetic Algorithm</td>
<td>TDM</td>
<td>Time Division Multiple Access</td>
</tr>
<tr>
<td>IIoT</td>
<td>Industrial Internet of Things</td>
<td>UAV</td>
<td>Unmanned Aerial Vehicle</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
<td>UE</td>
<td>User Equipment</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
<td>VEC</td>
<td>Vehicle Edge Computing</td>
</tr>
<tr>
<td>ITS</td>
<td>Intelligent Transportation Systems</td>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long Short-Term Memory</td>
<td>WAN</td>
<td>Wireless Access Network</td>
</tr>
</tbody>
</table>

---

**Fig. 3.** Road map of the survey.
three layers. Then, we elaborate on four different collaborations for resource scheduling under the three-tier architecture, i.e., things-edge collaboration, things-edge-cloud collaboration, edge-edge collaboration, and edge-cloud collaboration, as shown in Fig. 5.

1) Thing Layer: The thing layer, also known as the user layer, is composed of various end-devices (a.k.a., things), such as UAVs [40], CAVs [16], AR equipment [41], surveillance cameras for smart city [42], sensors for smart health [43], IoT devices for smart manufacturing [44], [45], smart devices for smart home [46]. In different works, end-devices are also called MDs or mobile users (MUs). Various things can perceive and have certain storage and computing capability. Things continuously generate and collect multiple types of data. Based on the QoE and QoS requirements of things, the data can be processed locally, or be offloaded to the edge and the cloud. In the edge computing network, there are not only static end-devices (e.g., sensors in smart homes, video cameras in public places) but also dynamic ones such as UAVs and vehicles, making the resource management even more challenging. Therefore, different solutions are proposed to address this issue, which are discussed in Section IV.

2) Edge Layer: The edge layer, as the core of the three-tier architecture, is an intermediate layer between the thing layer and the cloud layer. From the perspective of hardware composition, the edge layer consists of various networking and computing equipment, such as cellular tower, edge server (ES), roadside unit (RSU), gateway, edge controller, etc. The edge layer provides wireless access to smart devices through the radio access technology, such as Long Term Evolution (LTE), Wireless Fidelity (WiFi), and Dedicated Short-Range Communications (DSRC). Basically, the edge layer can provide more powerful storage and computing capabilities than the thing layer. From the perspective of software composition, the edge layer has edge management capabilities that offer service orchestration and invocation and schedule the ESs to

Fig. 4. Architecture for Resource Scheduling in Edge Computing.

Fig. 5. Four different collaboration manners for resource scheduling under three-tier architecture.
complete tasks. The edge layer can receive, process, and forward data streams from the thing layer, and achieve intelligent sensing, privacy protection, data analysis, intelligent computing, process optimization, and real-time control. Besides, since the edge and the cloud are complementary and mutually reinforcing, services in the cloud can be offloaded to the edge layer for load balancing and better QoE. With the objective of reducing bandwidth usage and energy consumption of the CN as well as reducing the communication overhead between the edge and the cloud, the edge layer is expected to schedule edge resources to enable rapid service response.

3) Cloud Layer: The cloud layer consists of the existing cloud computing infrastructures, such as computing units, storage units, and micro data centers (MDCs), connected with the edge layer through the CN (a.k.a, backbone network). Among the three layers, the cloud layer is undoubtedly the most powerful data processing and storage center. While ESs in the edge layer can process large amounts of data to reduce latency and energy consumption, the edge computing paradigm still requires the computing power and high-capacity storage infrastructure of the cloud to handle some tough tasks and global information. For example, the cloud layer can receive data streams from the edge layer, and send control information to the edge layer, and then from the edge layer to the thing layer, thereby optimizing the resource scheduling and field production process from a global perspective. Besides, based on the network resource distribution, the cloud layer can also dynamically adjust the deployment strategies and algorithms. Furthermore, it also provides decision-support systems, intelligent production, networking collaboration, service extension, personalized and customized service, and other domain-specific application services.

B. Things-Edge Collaboration

The resource scheduling in a things-edge collaboration manner involves the things layer and the edge layer. The task generated from smart devices can be processed locally or offloaded to ESs. Whether to offload these data depends on the things-edge collaboration strategy and the QoS and QoE requirements of smart devices. For example, Ali et al. in [47] proposed to select an optimal set of computation components to offload to ESs, aiming at minimizing the energy consumption of MDs. In addition to offloading task to the ES in a local region, Wang et al. in [48] proposed that the task can also be offloaded to the ES in a nearby region to reduce overall system costs and guarantee users’ QoE. Since the service requests of MUs and location may be dynamically changing, the static ES deployment may cause a “service hole.” To compensate for this issue and to improve the resource utilization as well as the system utility, Liu et al. in [49] explored a vehicle edge computing (VEC) network architecture and regarded the moving vehicles as vehicular ESs to assist the fixed ES to process the task from MUs. Besides, regarding UAVs as ESs is also a research trend. Yang et al. in [50] considered a UAV-enabled mobile edge computing (MEC) network, where the computation tasks from MUs can be processed by UAVs aiming at minimizing the power consumption of all MUs and UAVs. Unlike previous studies in which users first offload task to ES and results are then fed back, Chen et al. in [51] investigated the relay-assisted computation offloading (RACO). In the considered RACO scenario, a mobile-edge relay server (MERS) is utilized to assist the results of computational tasks among users by allocating computing and communication resources.

C. Things-Edge-Cloud Collaboration

Although the things-edge collaboration manner has a relatively powerful capacity, it ignores the huge computing resources in the cloud computing center. With the ever-increasing smart devices and their resource-hungry applications, it will become increasingly difficult to rely on the resources in the edge layer alone to meet the service requirements of smart devices. Therefore, it is particularly important and necessary to take full advantage of both edge computing and cloud computing and make them complementary to design a collaborative paradigm, the things-edge-cloud collaboration manner. Guo and Liu in [52] introduced the concept of a hybrid fiber-wireless (FiWi) network, in which the multi-access edge computing and the centralized cloud computing cooperated to provide better offloading performance and good scalability as computation tasks increase. The combination of edge computing and cloud computing FiWi takes the complementary advantages of good scalability, high mobility, and supports diverse wireless access technologies in edge computing, large capacity, high reliability, and low-latency in fiber-enabled cloud computing. For the resource-intensive applications, such as big-data analytics, AI processing, and 3D sensing from industrial Internet of things (IoT) devices, Hong et al. in [53] proposed a multi-hop IIoT-edge-cloud collaborative computation offloading paradigm, aiming at minimizing energy consumption and computing time of task processing. Wang et al. in [54] proposed the concept of “HetMEC”, which refers to heterogeneous multi-layer MEC. In HetMEC, if the task offloaded from smart devices cannot be processed on time by the ES, it can be offloaded to the cloud center, aiming at minimizing transmission and computing time. Different from previous studies, Dinh et al. in [14] considered renting computing resources termed virtual machines (VMs) from the cloud layer to scale up the capacity of the edge layer, with the goal of minimizing the total cost, including the processing cost at the edge, the remote on-demand VMs cost, the reserving and using remote reserved VMs cost.

D. Edge-Edge Collaboration

Generally, the edge-edge collaboration manner for resource scheduling in edge computing does not arise in isolation. Instead, it usually comes along with the things-edge collaboration manner or the things-edge-cloud collaboration manner. Through an edge-edge collaboration manner, there is one more option for task processing. Many studies have investigated this collaboration manner. Huang et al. in [36] proposed a parked vehicle edge computing (PVEC) architecture, where idle resources of PVs can be fully utilized. In PVEC architecture, VEC servers explore opportunistic resources from PVs to
allocate workloads, and provide rewards to PVs for their assistance. When necessary, VEC servers can also undertake the residual workloads. As a result, VEC servers and PVs cooperate to process task in an edge-edge collaboration manner. To alleviate the workload on ESs, Na et al. in [55] proposed to utilize edge gateways (EGs) at the edge layer to assist task processing. A resource orchestration scheme among EGs and/or between ES and EGs is also proposed, aiming to maximize the efficiency of IoT systems. Alameddine et al. in [56] studied the dynamic task offloading and scheduling problem (DTOS) in multi-access edge computing, where application’s task assignment and the order of execution are jointly considered. The tasks that cannot be processed by its corresponding eNB-enabled ES can be offloaded to another ES in an edge-edge collaboration manner to meet UE’s QoE requirement. Miao et al. in [57] proposed an intelligent offloading strategy based on the mobile-edge cloud computing architecture, where tasks are scheduled among MDs, ESs, and the cloud based on task prediction, aiming at reducing the total task delay. Besides, the ES in this strategy can decide whether to migrate its overload to other ES in an edge-edge collaboration manner. Differently, Thai et al. in [58] proposed a cloud-edge computing architecture to provide horizontal and vertical collaborations, aiming to minimize the total cost. Horizontal collaboration means that offloading operations can be conducted among the nodes in the same tier, while vertical collaboration means that offloading operations can be conducted among the cross-tier nodes.

E. Edge-Cloud Collaboration

If most computing tasks are performed in the cloud computing center in the considered three-tier architecture, long latency will be produced, which can not satisfy users’ QoE. The long latency problem can be improved by offloading some or all of the tasks in the cloud center to the edge in an edge-cloud collaboration manner, such as the edge accelerated Web platform (EAWP) by Nippon Telegraph and Telephone Corporation [61]. The edge-cloud collaboration manner can be used in many applications. For example, mobile client shopping has become popular where customers frequently operate the shopping cart. The change of the shopping cart status is first completed in the cloud center, and then the product view is updated on the MD, which results in long latency. If shopping cart data can be cached and relevant actions can be performed on the edge, the new product view will be pushed to the MD once the customer’s request reaches the edge, thus greatly improving the customer’s QoE. Another example is the video transcoding application. Online video traffic on MDs is growing exponentially in network traffic [62], [63], and MUs have high QoE requirements for streaming video. The video transcoding has become an optimized technique for video data transmission. However, since video transcoding consumes a great quantity of computing and storage resources, it is typically executed in the offline media server (located in the cloud layer). Unfortunately, this approach may increase the latency when the video stream is redirected from the media server and the real-time streaming service cannot be provided. To this end, Yoon et al. in [15] proposed to run the video transcoding on ENs such as home WiFi access point. The experimental results show that their solution is low-cost, transparent, and scalable. Besides, Xu et al. in [59] proposed to regard the edge layer as MDCs to provide edge computing services. A model, named Zenith, was also proposed, where SPs can establish resource sharing contracts with edge infrastructure providers, aiming to increase resource utilization and minimize job execution latency. Similarly, Zhang et al. in [60] proposed to deploy SPs in the edge layer to manage the task processing for MUs. The SPs can schedule the task to the edge or the cloud in an edge-cloud collaboration manner, aiming at providing high-quality services and maximizing the total profit of all SPs.

For simplicity, a comparison of papers focusing on different collaboration manner for resource scheduling are summarized in Table III.

III. BASIC MODEL AND RESEARCH ISSUES

In this section, we first present the basic model for resource scheduling in edge computing, which guides users to decide whether to take offloading action based on the current communication and computing resource state as well as their QoE requirements. Then, we elaborate on the state-of-the-art research on resource scheduling in edge computing from three aspects: computation offloading, resource allocation, and resource provisioning.

A. Basic Model

In a typical edge computing scenario, various tasks would be generated from user devices. Generally, an arbitrary task $T$ can be described by five items, i.e., $T = (D, c, \alpha, \gamma, \tau)$, where $D$ is the data size of $T$, $c$ represents the processing density (in CPU cycles/bit) of $T$, $\alpha$ ($0 \leq \alpha \leq 1$) stands for the parallelizable fraction of $T$, $\gamma$ denotes the ratio of the data size of processing result to the data size of $T$, and $\tau$ represents the delay constraint of $T$ [10]. The end-devices, CAVs and UAVs, can be connected to the edge through various communication channels (such as 4G/5G, WiFi, LTE/DSRC, etc.) We denote the wireless bandwidth assigned to the end-devices for task $T$ as $B$. The generated task $T$ can be processed locally or offloaded to the edge or the cloud to be processed. The offloading action is taken based on different requirements for energy consumption, latency, cost, and computing acceleration. Let $\lambda$ ($0 \leq \lambda \leq 1$) denote the offloading decision variable, which represents the ratio of the offloaded data size to the total data size of task $T$. If $\lambda = 0$, task $T$ will be processed locally; if $\lambda = 1$, task $T$ will be fully offloaded; otherwise, the data with size $\lambda D$ will be offloaded, the data with size $(1 - \lambda)D$ will be processed locally. In the following, we will demonstrate the local processing part and offloading part, respectively.

1) Task $T$ Processed Locally: The number of cores of the users is denoted as $n_1$, and the processing capability (i.e., the amount of CPU frequency in cycles/s) of each core assigned for local computing as $f^l$, then the power consumption of each core for a user to process data locally is expressed as $p^l = \kappa_1(f^l)^3$, where $\kappa_1$ is a coefficient reflecting the relationship
TABLE III

Comparison of papers focusing on different collaboration manner for resource scheduling. Acronyms used in this table:

- USER EQUIPMENT (UE), EDGE SERVER (ES), MOBILE DEVICE (MD), VEHICULAR EDGE SERVER (VES), FIXED EDGE SERVER (FES), MOBILE EDGE RELAY SERVER (MERS), Base Station (BS), UNMANNED AERIAL VEHICL (UAV), EDGE GATEWAY (EG), PARKED VEHICLE (PV), MOBILE USER (MU), MICRO DATA CENTER (MDC)

<table>
<thead>
<tr>
<th>Paper</th>
<th>Collaboration Manner</th>
<th>Things</th>
<th>Edge</th>
<th>Research Issue</th>
<th>Characteristics</th>
<th>Methodology</th>
</tr>
</thead>
<tbody>
<tr>
<td>[47]</td>
<td>Things-edge</td>
<td>UE</td>
<td>ES</td>
<td>Offloading strategy</td>
<td>Minimize the energy consumption of MDs by selecting an optimal set of computation components to offload to ESs.</td>
<td>Deep learning</td>
</tr>
<tr>
<td>[48]</td>
<td>Things-edge</td>
<td>UE</td>
<td>ES in local and nearby region</td>
<td>Offloading strategy</td>
<td>Formulate the computation offloading problem as a potential game</td>
<td>Game theory, Jacobi algorithm</td>
</tr>
<tr>
<td>[49]</td>
<td>Things-edge</td>
<td>UE</td>
<td>VES and FES</td>
<td>Offloading strategy; resource allocation</td>
<td>Consider the stochastic vehicle traffic, dynamic computation requests and time-varying communication conditions</td>
<td>Reinforcement learning</td>
</tr>
<tr>
<td>[50]</td>
<td>Things-edge</td>
<td>UE</td>
<td>UAV</td>
<td>Resource allocation</td>
<td>Jointly optimize user association, power control, computation capacity allocation and location planning</td>
<td>Compressive sensing, search method</td>
</tr>
<tr>
<td>[51]</td>
<td>Things-edge</td>
<td>User</td>
<td>MERS</td>
<td>Computation offloading; resource allocation</td>
<td>Jointly optimize transmit powers, processor speeds, bandwidth, and offloading ratio</td>
<td>Iterative algorithm</td>
</tr>
<tr>
<td>[52]</td>
<td>Things-edge-cloud</td>
<td>MD</td>
<td>ES</td>
<td>Offloading strategy</td>
<td>Minimize all MDs’ energy consumption while satisfying the MDs’ computation execution time constraint</td>
<td>Game theory</td>
</tr>
<tr>
<td>[53]</td>
<td>Things-edge-cloud</td>
<td>IoT devices</td>
<td>BS enabled ES</td>
<td>Offloading strategy</td>
<td>Minimize energy consumption and computing time of task processing</td>
<td>Game theory</td>
</tr>
<tr>
<td>[54]</td>
<td>Things-edge-cloud</td>
<td>User</td>
<td>ES</td>
<td>Resource allocation</td>
<td>Consider the edge’s local processing cost and capacity, the cloud’s multiple rental options</td>
<td>Offline and online algorithms</td>
</tr>
<tr>
<td>[55]</td>
<td>Things-edge-cloud</td>
<td>IoT devices</td>
<td>EG; ES</td>
<td>Resource allocation</td>
<td>The communication and computing resources, the task assignment among multiple layers are jointly coordinated</td>
<td>Latency minimization algorithm</td>
</tr>
<tr>
<td>[56]</td>
<td>Things-edge-edge</td>
<td>Mobile vehicles</td>
<td>PVs; VES</td>
<td>Resource allocation</td>
<td>Consider computing capacities of BSs and EGS, and interference among EGSs</td>
<td>Lagrangian and KKT condition</td>
</tr>
<tr>
<td>[57]</td>
<td>Things-edge-edge-edge</td>
<td>UE</td>
<td>eNB enabled ES</td>
<td>Computation offloading; resource allocation</td>
<td>The tasks from UEs is scheduled among different ESs</td>
<td>Benders decomposition technique</td>
</tr>
<tr>
<td>[58]</td>
<td>Things-edge-edge-edge</td>
<td>MD</td>
<td>ES</td>
<td>Computation offloading</td>
<td>Integrate artificial intelligence (AI), local computing, edge computing, and cloud computing</td>
<td>Deep learning, LSTM</td>
</tr>
<tr>
<td>[59]</td>
<td>Things-edge-edge-edge</td>
<td>Smart device</td>
<td>ES</td>
<td>Computation offloading</td>
<td>Vertical and horizontal offloading: workload and capacity optimization problem</td>
<td>Branch-and-bound method</td>
</tr>
<tr>
<td>[60]</td>
<td>Things-edge-edge-edge</td>
<td>MU</td>
<td>ES</td>
<td>Resource placement</td>
<td>Place the video transcoding function at edge layer, provide higher video bit-rates without causing video stall or buffering</td>
<td>Video transcoding at edge</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Smart device</td>
<td>MDC</td>
<td>Resource allocation; resource provisioning</td>
<td>SPs put resource in the edge layer; a latency-aware task scheduling mechanism</td>
<td>Auction-based contracts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UE</td>
<td>ES</td>
<td>Resource allocation</td>
<td>SPs at the edge layer assigns the tasks from UEs to be processed in base station or cloud center</td>
<td>Decentralized multi-SP resource allocation</td>
</tr>
</tbody>
</table>

between processing capability and power consumption at the end-device side [64].

Local computing time: Based on the Amdahl’s law [65], the local computing time for \((1 - \lambda)D\) bits data of the task, which consists of the computing time of the serialized part \(t^s = c(1 - \alpha)(1 - \lambda)D/f^s\) and the computing time of the parallelizable part \(t^p = co(1 - \lambda)D/f^p n_1\), can be calculated as

\[
t^l = t^s + t^p = \frac{c(1 - \lambda)D}{f^l} \left(1 - \alpha + \frac{\alpha}{n_1}\right).
\]

Local energy consumption: The energy consumption for local computing is formulated as

\[
E^l = p^l t^l + n_1 p^l t^p = k_\lambda c D(1 - \lambda) \left(\frac{f^l}{1 - \lambda}\right)^2.
\]

2) Task T Offloaded to the Edge: The data of task T can be offloaded to the edge through wireless communication links. For the data transmission rate, we use \(r\) to denote it. The data transmission rate can be characterized by various wireless transmission models based on Shannon’s formula. For example, Wang et al. in [66] model the path loss as \(d^{-\vartheta}\), where \(d\) denotes the distance from the end-device to the edge, and \(\vartheta\) denotes the path loss exponent. Based on Shannon’s formula, when data is offloaded from the end-device to the edge over the assigned wireless bandwidth \(B\), the transmission rate can be expressed as

\[
r_1 = B \log_2\left(1 + \frac{P_1 |h|^2}{\omega_0 d^\vartheta}\right),
\]

where \(P_1\) is the transmission power of the end-device, \(h\) is the channel fading coefficient, and \(\omega_0\) denotes the white Gaussian noise power.
Transmission delay for offloading: Based on the analysis above, the transmission delay for offloading $\lambda D$ bits of data to the edge can be obtained by

$$t_{\text{up}} = \frac{\lambda D}{r_1}. \quad (3)$$

Transmission energy consumption for offloading: Accordingly, the energy consumption of the end-device for transmitting the offloaded $\lambda D$ bits of data is expressed as

$$E_{\text{up}} = P_1 t_{\text{up}} = \frac{\lambda D P_1}{r_1}. \quad (4)$$

Computing time at the edge: After the $\lambda D$ bits of data is offloaded to the edge, the edge would process the data. Let $n_2$ denote the number of cores assigned for task processing of the edge, $f^e$ denote the processing capability (i.e., the amount of CPU frequency in cycles/s) of each core ($f^e \gg f^l$). The power consumption of each core of the edge to process data can be expressed as $p^e = \kappa_2 (f^e)^3$, where $\kappa_2$ is a coefficient reflecting the relationship between processing capability and power consumption on the edge side [64]. And the computing time of the offloaded $\lambda D$ bits of data, which consists of the computing time of the serialized part $t^e_s = e (1 - \alpha) D / f^e$ and the computing time of the parallelizable part $t^e_p = c \lambda \alpha D / n_2 f^e$, can be formulated as

$$t^e = t^e_s + t^e_p = \frac{e \lambda D}{f^e} \left( 1 - \alpha + \frac{\alpha}{n_2} \right). \quad (5)$$

Energy consumption at the edge: The energy consumption of the edge for computing the $\lambda D$ bits of data is formulated as

$$E^e = p^e t^e_s + n_2 p^e t^e_p = \kappa_2 c D (f^e)^2. \quad (6)$$

3) Result Return: After the task $T$ has been processed, the result will be returned to the end-device. Generally, the return process has been neglected in many works since the processing result is usually very tiny [67]–[69]. As a general model, we will consider the return result process. Let $r_2$ denote the data transmission rate in the result return process, then similar to the offloading data rate, $r_2$ can be formulated as

$$r_2 = B \log_2 (1 + \frac{P_2 |A|^2}{\omega_0 \sigma_D^2}), \quad (11)$$

Transmission delay for result return: Based on the analysis above, the transmission delay for $\gamma D$ bits result return can be obtained by

$$t_{\text{down}} = \frac{\gamma D}{r_2}. \quad (7)$$

Transmission energy consumption for result return: Accordingly, the energy consumption of the EN for transmitting the $\gamma D$-bits of processing result to the end-device is expressed as

$$E_{\text{down}} = P_2 t_{\text{down}} = \frac{\gamma D P_2}{r_2}. \quad (8)$$

4) Total Delay: Based on the analysis above, the total delay of processing task $T$ is a combination of local computing time, transmission delay for offloading, computing time at the edge, and transmission delay for result return, which is formulated as

$$t = \min \left\{ t^l, t_{\text{up}} + t^e + t_{\text{down}} \right\}. \quad (9)$$

5) Total Cost: The total cost of processing task $T$ comes from three aspects, including energy consumption, use of bandwidth resources, and use of computing resources. For the energy consumption, let $\phi$ denote the weight coefficient that indicates the energy consumption cost of one unit energy during task computing and transmitting [70], then the energy consumption cost can be formulated as

$$C_{\text{energy}} = \phi (E^l + E^{tr} + E^e + E_{\text{down}}). \quad (10)$$

For the bandwidth cost, let $p_1$ denote the cost of using per unit of bandwidth per unit of time, the bandwidth cost can be formulated as

$$C_{\text{comm}} = p_1 B (t_{\text{up}} + t_{\text{down}}). \quad (11)$$

For the computing cost, let $p_2$ denote the cost of using per unit of processing capability per unit of time, then the computing cost can be formulated as

$$C_{\text{comp}} = p_2 n_2 f^e t^e. \quad (12)$$

Therefore, the total cost for processing task $T$ can be expressed as

$$C = C_{\text{energy}} + C_{\text{comm}} + C_{\text{comp}}. \quad (13)$$

6) Computing Acceleration: Before the task offloading decision is made, some other QoE requirement such as computing acceleration is also a key consideration. The computing acceleration refers to the speedup of processing a task at the edge when compared with computing it locally. According to Amdahl’s law, the speedup can be obtained if the $(1 - \lambda) D$ bits of task data is computed locally as follows,

$$S_1 = \frac{1}{(1 - \alpha) + \frac{\alpha}{n_2}}. \quad (14)$$

Similarly, the speedup can be obtained if the $\lambda D$ bits of task data is computed at the edge by the following formula,

$$S_2 = \frac{1}{(1 - \alpha) + \frac{\alpha}{n_2}}. \quad (15)$$

However, when task data is offloaded to the edge for processing, the actual latency comes from computing delay and transmission delay. In this circumstance, the actual computing acceleration is expressed as

$$A = \frac{t^l}{t_{\text{up}} + t^e + t_{\text{down}}}. \quad (14)$$

According to the above basic model, many aspects should be considered to achieve the different requirements of both end-devices and the system for energy consumption, latency, cost, and computing acceleration. The first aspect is to decide the offloading variable $\lambda$, i.e., an efficient computation offloading. The second aspect is to decide the variables $B, n_1, n_2, f^l, f^e$, i.e., resource allocation of the communication and computing resources. The third aspect is to decide the association between tasks and ENs and the placement of computing resources, i.e., resource provisioning. The outline of the three research issues is shown in Fig. 6 and is described in detail below.
B. Computation Offloading

The computation offloading is a very important research issue for resource scheduling in edge computing, which brings services to the proximity of data source [34]. This subsection reviews the research on this issue. As shown in Fig. 7, the computation offloading can be broadly classified on the base of: a) the direction of offloading, namely from device to edge, from edge to cloud, from cloud to edge, from device to device, and from edge to edge, and b) the granularity of offloading, namely binary offloading and partial offloading.

1) Direction: Since end-devices in the thing layer are mostly resource-constrained, resource-intensive tasks need to be fully or partially offloaded to ENs with powerful computing resources. The computation offloading from end-devices to ENs compensates for the deficiency of end-devices in computing performance, storage, and energy efficiency. Also, the computation offloading from end-devices to ENs can alleviate the overload of the cloud computing center and reduce the delay caused by wireless transmission. For example, video data from surveillance cameras can be offloaded to the EN for low-delay and privacy-protecting analysis and process, compared with being offloaded to the cloud computing center. In addition, the upward offloading has also promoted the development of the super low-delay applications such as video services and CAVs. The application data of real-time perception need to be offloaded to ENs for rapid processing, which guides vehicles to take right driving actions. Similarly, if ENs are unable to process the task data offloaded from end-devices in a timely manner, it can be offloaded to the cloud center. The computation offloading ways both from end-devices to ENs and from ENs to the cloud center can be referred to as upward offloading.

The computation offloading also concentrates on downward offloading, which means the offloading from the cloud center to the edge. In the edge-cloud collaboration manner discussed in the last section, this kind of offloading is adopted. Both upward offloading and downward offloading are regarded as vertical offloading. In addition to vertical offloading, the computation offloading manner also includes horizontal offloading. There are two research issues in horizontal offloading. The first one is that end-devices can offload their resource-intensive tasks to other end-devices with idle computing resources. The second is that one EN can also migrate their task data to other ENs for processing. Thus, there are in total five different offloading directions in the vertical offloading and horizontal offloading, which will be discussed in the following.

a) Device-to-edge: For applications that require powerful capacity or edge data aggregation, various end-devices will offload their tasks to ENs. This offloading direction is the focus of computation offloading, and it is operated under the things-edge collaboration manner as discussed in Section II-B. The offloading from end-devices to ENs can achieve different QoS and QoE requirements for end-devices. For example, for reducing the task processing latency, Chen et al. in [87] considered to offload the computation tasks from MDs to small-cell base stations (BSs) with cloud-like computing and storage capabilities, with the aim of minimizing the long-term system delay. For reducing energy consumption, Guo et al. in [88] proposed to offload the computation tasks from MDs to small BSs, and an efficient computation offloading scheme by jointly considering offloading decision-making and resource allocation was proposed, aiming at reducing the energy consumption of MDs. Also, Guo et al. in [88] considered an ultra-dense edge computing network, where MDs’ energy consumption is minimized by offloading their tasks to ENs. Besides, Jošilo and Dán in [89] proposed a computation offloading scheduling scheme to determine whether to offload the tasks of end-devices to ENs, aiming to minimize the cost that is a combination of delay and energy consumption.

b) Edge-to-cloud: Generally, the tasks offloaded from end-devices are processed by computing nodes in the edge layer. The computing nodes, including cloudlets, ENs, BSs, mini data centers, etc., can provide different capacities. If the task data in the edge layer cannot be processed by the computing node in time, they can be further offloaded to the cloud center to achieve a balanced overload. This kind of offloading direction, from the edge to the cloud, is actually operated under the edge-cloud collaboration manner, as discussed in Section II-E. For example, in the area of CAVs, Zhang et al. in [90] proposed to improve the system utility by utilizing a multi-level offloading scheme among ENs and cloud servers. Also, Zhao et al. in [91] considered to jointly optimize the offloading decision and resource allocation by an edge-cloud collaborative offloading approach.

c) Cloud-to-edge: This kind of offloading direction is also operated under the edge-cloud collaboration manner as
discussed in Section II-E, which brings computation tasks from the distant cloud to the edge to achieve lower data transmission latency, thereby shortening the application response time. The typical issues of the cloud-to-edge offloading mainly include: (i) video transcoding on ENs [15]; (ii) application cloning from cloud to edge to provide users with better QoE [92]; (iii) data replication on the edge [59], [93]–[95]; (iv) edge discovery and management, where workloads are offloaded from the cloud to the chosen ENs and the orchestration across multiple ENs is evaluated [96], [97].

d) Edge-to-edge: The edge-to-edge offloading is actually operated under the edge-edge collaboration manner, as discussed in Section II-D, which can alleviate the workload of some overloaded EN by offloading (or migrating) some workloads to a peer. The typical issues of the edge-to-edge offloading mainly include: (i) task scheduling, which can orchestrate the task processing among different ENs [55]–[58]; (ii) service migration, by which services are dynamically migrated across multiple heterogeneous ENs [98], [99]; (iii) offload forwarding, in which an EN is regarded as a relay to forward workloads to neighboring ENs [100].

e) Device-to-device: The device-to-device offloading can be operated under both the things-edge collaboration manner and the things-edge-cloud collaboration manner, as discussed in Section II-B and Section II-C, which offloads the workloads from one end-device to a peer by making full use of idle resources. For example, Luo et al. in [101] proposed a collaborative task data scheduling scheme in VEC, where the computation tasks of vehicles can be not only processed locally, i.e., offloaded to RSUs, but also can be migrated to other vehicles with idle computing resources.

2) Granularity: As one of the important research issues in computation offloading, the offloading decision-making problem focuses on whether and how much to offload. Depending on whether the computation task is dividable or not, the granularity of offloading can be classified into two categories: a) binary offloading, and b) partial offloading, which will be presented in the following.

a) Binary offloading: Binary offloading, also known as “0-1 offloading”, means the whole computation task is either processed locally or offloaded to elsewhere. “0” and “1” are the indicators of whether the task is offloaded or not. Generally, “0” means the whole task is processed locally, and “1” means it is offloaded to elsewhere [53], [75]. When the whole task is processed locally, the computing time, energy consumption, and the cost of processing task are determined by the local capacity. When the whole task is offloaded to other nodes to process, the computing time mainly includes task transmission time and task processing time. Similarly, energy consumption mainly includes transmission energy consumption and processing energy consumption. The cost mainly includes transmission cost and processing cost. From this point of view, the factors that affect the offloading performance include wireless channel conditions, wireless bandwidth, and processing capability of the destination node (i.e., the node to which the task is offloaded). The research on binary offloading involves in the association between tasks and destination nodes [102]–[106], which refers to the determination of the offloading of a specific task to a destination node, among various tasks and destination nodes.

b) Partial offloading: Partial offloading allows flexible components/data partitioning, which means that a task can be divided into separated parts [17], [107]–[109]. The research on partial offloading is to determine how much and in what way of the whole task can be offloaded to the destination node. Generally, a ratio known as “offloading ratio” is set to indicate the proportion of offloading part of the task. Partial offloading involves two parts of task processing, the local processing part and the offloading part. Accordingly, the task processing performance is jointly determined by the computing time, energy consumption, and the cost of processing task locally and at the destination side. Actually, in addition to deciding and optimizing the offloading ratio to achieve various QoS requirements, the study of partial offloading also involves in the association between the offloading part of the task and the destination node [110].

In most existing works, neither binary offloading or partial offloading issues can be addressed alone, and other issues such as resource allocation [111]–[114] and resource provisioning [115], [116] are jointly studied with computation offloading, which will be presented in later sections. To enable readers to grasp basic ideas of computation offloading on both binary offloading and partial offloading, a comparison of papers focusing on this research issue is presented in Table IV.

C. Resource Allocation

As another important research issue in resource scheduling, resource allocation studies how to reasonably and effectively allocate resources in the edge computing system to complete offloading and task processing. Generally, the main resources involved in the current research on resource allocation are computing, communication, and storage resources. Computing resources typically refer to CPU cycles and resource blocks (VMs/containers). Communication resources refer to wireless resources including bandwidth, spectrum, power, and link used for data transmission during computation offloading. Storage resources are used to cache computation tasks and popular content (e.g., on-demand video, AR/VR, road surveillance, etc.) to the edge of the network, reducing the service response time and the burden on the network. Some research on resource allocation only focuses on allocating one kind of resource while most research considering the joint resource allocation, which will be elaborated on in the following.

1) Single Resource: The existing works involved in the single-resource allocation mainly focus on the allocation of computing or communication resources. In the computation offloading decision-making problem, many works consider the allocation of communication resources. Like the works in [117] and [118], both focused on communication resources and studied how to allocate the transmission power during the offloading process, with the goal of minimizing the system’s energy consumption. Differently, Li et al. in [119] studied the channel selection for task offloading. The effect of multi-channel interference on the energy efficiency of task offloading was taken into account. Obviously, the most important thing in
the offloading process is the allocation of computing resources. The work in [120] designed the selective offloading scheme for IoT devices, and it studied how to allocate the best EN for offloading tasks to minimize energy consumption. Similarly, Xu et al. in [121] studied the computation offloading problem for IoT-enabled cloud-edge computing, and they focused on how to allocate the computing resource for tasks to minimize the execution time and energy consumption for MDs. Also, some studies only consider storage resources in terms of caching data [122] and caching service [123], [124]. Yu et al. in [122] proposed a collaborative offloading with data caching enhancement strategy to minimize the total delay. Caching services such as databases or libraries on ENs for task execution can effectively reduce the total delay. The study in [123] focused on dynamic service caching and task offloading, and proposed an online algorithm based on Lyapunov optimization and Gibbs sampling.

2) Computing and Communication (CC): The offloading process often involves the joint allocation of communication and computing resources. Many existing works have studied this topic [125]–[134]. Guo et al. in [126] proposed an adaptive resource allocation framework for MEC, which applied the idea of blockchain into the framework design. They formulated an optimization problem for spectrum and block allocation. The study in [127] formulated the problem of optimizing the joint allocation of computing resources on ENs and radio resources under the non-orthogonal multiple access (NOMA) protocol and used an efficient layer algorithm to solve it. Likely, to maximize the total revenue, Wang et al. in [129] studied the optimization problem for bandwidth and computation allocation with the QoS-guaranteed constraint, and they proposed an algorithm based on alternating direction method of multipliers (ADMM) to solve it. Under the transmission protocol of time division multiple access (TDMA), the authors in [130] studied how to assign the time and rate of local users for task offloading and how to allocate computation frequency for task execution, aiming to minimize the computation latency. Similarly, the work in [131] also adopted TDMA transmission protocol. Millimeter-wave (mmWave) communication as one of the promising transmission protocols was applied in the work [132]. This paper formulated the joint beamforming vectors at the users and computation ratios at ENs allocation problem to minimize the system delay, and proposed a penalty dual decomposition technique to solve this optimization problem.

3) Computing, Communication, and Storage (CCS): Many works have considered communication, computing, and storage resources simultaneously in the resource allocation problem [8], [67], [135]–[137]. In recent years, the prevalence of edge intelligence has attracted widespread attention from academia and industry. In the work [135], the authors designed an In-Edge AI framework for optimizing computing, communication, and caching allocation. They utilized both deep reinforcement learning and federated learning (FL) techniques to optimize the edge system’s performance. Liang et al. in [136] studied the bandwidth provisioning and content source selection problem by introducing caching and computing functions in MEC. They proposed a decentralized approach based on ADMM to solve it. Likely, the work in [67] addressed the optimization problem for joint computation offloading, resource allocation, and content caching, in which computing, spectrum, and caching resources were considered simultaneously. Particularly, all resources in the study [8] were in the form of virtual resources. The authors formulated a joint virtual resource (including spectrum, caching, and computing) allocation problem, intending to maximize the system’s utility. Similarly, the authors in [137] also studied the virtual resource allocation problem in which the communication, computation, and caching resources can be shared among all users. Besides, they presented a distributed algorithm based on ADMM to

<table>
<thead>
<tr>
<th>Gran.</th>
<th>Paper</th>
<th>Objective</th>
<th>Research Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Offloading</td>
<td>[71]</td>
<td>Delay, energy consumption</td>
<td>a) Offloading decision; b) transmission power allocation; c) CPU frequency allocation;</td>
</tr>
<tr>
<td></td>
<td>[72]</td>
<td>Utility</td>
<td>a) Offloading proportion determining; b) power allocation; c) energy harvesting;</td>
</tr>
<tr>
<td></td>
<td>[73]</td>
<td>Energy consumption</td>
<td>a) Task-destination association; b) offloading decision;</td>
</tr>
<tr>
<td></td>
<td>[74]</td>
<td>Energy consumption</td>
<td>a) Task-destination association; b) offloading decision; c) task ready time determining;</td>
</tr>
<tr>
<td></td>
<td>[75]</td>
<td>Utility</td>
<td>a) Task-destination association; b) offloading decision;</td>
</tr>
<tr>
<td></td>
<td>[76]</td>
<td>Energy consumption</td>
<td>a) Transmission power allocation; b) offloading decision; c) CPU clock allocation;</td>
</tr>
<tr>
<td></td>
<td>[77]</td>
<td>Latency, energy consumption</td>
<td>a) Task-destination association; b) wireless channel allocation; c) computation capability allocation;</td>
</tr>
<tr>
<td></td>
<td>[78]</td>
<td>Energy consumption</td>
<td>a) Task-destination association; b) computing capability allocation;</td>
</tr>
<tr>
<td>Partial Offloading</td>
<td>[79]</td>
<td>Revenue</td>
<td>a) Task-destination association; b) offloading workload amount determining; c) energy harvesting;</td>
</tr>
<tr>
<td></td>
<td>[80]</td>
<td>Delay, energy consumption</td>
<td>a) Computing resource allocation; b) offloading ratio determining;</td>
</tr>
<tr>
<td></td>
<td>[81]</td>
<td>Latency</td>
<td>a) Task-destination association; b) offloading ratio determining;</td>
</tr>
<tr>
<td></td>
<td>[82]</td>
<td>Delay</td>
<td>a) Task-destination association; b) offloading decision;</td>
</tr>
<tr>
<td></td>
<td>[83]</td>
<td>Energy consumption</td>
<td>a) Offloading data amount determining; b) transmission power allocation; c) transmission time allocation;</td>
</tr>
<tr>
<td></td>
<td>[84]</td>
<td>Latency</td>
<td>a) Subcarrier assignment; b) offloading ratio determining; c) transmission power allocation;</td>
</tr>
<tr>
<td></td>
<td>[85]</td>
<td>Execution time</td>
<td>a) Subtask placement; b) topology/schedules of the IoT tasks;</td>
</tr>
<tr>
<td></td>
<td>[86]</td>
<td>Latency, resource utilization</td>
<td>a) Task placement; b) VM instance provisioning;</td>
</tr>
</tbody>
</table>
address the formulated problem. Moreover, a few research focus on joint communication and storage resource allocation problems [138], [139].

A comparison of papers focusing on resource allocation is presented in Table V. It can be observed that communication, computing, and storage resources are rarely allocated individually in resource scheduling. Many works combine two or three of them to model and jointly optimize the allocation simultaneously.

D. Resource Provisioning

Since loads of users’ requests vary over time, edge computing systems experience constant fluctuations in workload. These fluctuated workloads may cause problems such as over-provisioning or under-provisioning of edge resources. In the case of over-provisioning, where the resources allocated to some users are greater than the actual load demanded by users, the edge system may be unnecessarily costly. Besides, in under-provisioning, the resources allocated to users for the service are less than the actual load demanded by users, resulting in a poor QoS or even the inability to complete users’ tasks. Therefore, allocating the appropriate amount of edge resources to users dynamically to minimize the system cost and meet users’ QoS requirement is an important issue. Based on the analysis and summary of current research, the studies on resource provisioning in edge computing can be divided into two categories: a) task allocation, which is a passive resource provisioning from users’ perspective. The task allocation problem in edge computing refers to the optimal placement and matching plan between users’ tasks and edge resources; b) resource placement, which is an active resource provisioning from resource providers’ perspective. The resource placement mainly includes cloud service decentralization to the edge, optimized deployment of ESs, quantity allocation of edge resources, and virtual edge resource placement issues. In the following, we will elaborate on the two aspects.

1) Task Allocation: Yang et al. in [140] studied the cloudlet placement and task allocation problem. Then, they formed a mixed integer linear programming (MILP) problem and used the benders decomposition-based approach to solve it. Before task allocation, the authors investigated the resource placement, aiming to calculate the task delay and energy consumption of different ENs. It provides systematic conditions for task allocation. The work in [141] focused on data management in edge computing, and it presented a multi-layer scheduler considered the various context dimensions of data. In the multi-layer scheduler design, the tasks generated by data are allocated based on the current context and the system state during runtime. Fan et al. in [142] proposed a deadline-oriented task allocation mechanism and formed a task scheduling problem as a multi-dimensional 0-1 knapsack problem. They adopted an efficient task allocation algorithm based on ant colony optimization to increase the system’s total profit while satisfying the deadline and resource constraints of the task. There are some works on application placement, which focus on assigning tasks from users’ applications to the appropriate edge resources for processing [143]–[145]. It is essentially a task allocation problem. In [143], the authors designed a third-party platform responsible for allocating MUs’ application tasks to edge resource providers. MUs subscribe to the platform that collects the information of ENs to place tasks on ENs optimally. A programming algorithm was proposed to select the best task placement server from the users’ perspective to avoid task migration, thus minimizing the time cost. From the platform’s point, the efficient heuristic algorithm is presented to schedule tasks to minimize the total cost. Likely, Mahmud et al. in [144] proposed a QoE-aware scheme for application placement. The proposed scheme prioritized different tasks of applications and updated the capabilities of ENs according to their current status, thus facilitating optimal task allocation decisions. Later, for the edge-cloud environment, they proposed another application placement policy [145], aiming to maximize the edge system’s profit and ensure the user’s QoE.

<table>
<thead>
<tr>
<th>Paper</th>
<th>Computing</th>
<th>Communication</th>
<th>Storage</th>
<th>Algorithm</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>[117]</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>Majorization minimization method</td>
<td>Energy consumption</td>
</tr>
<tr>
<td>[118]</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>Genetic algorithm</td>
<td>Energy consumption</td>
</tr>
<tr>
<td>[119]</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>Auction-based approach</td>
<td>Energy consumption</td>
</tr>
<tr>
<td>[121]</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>NSGA-III algorithm</td>
<td>Delay, energy consumption</td>
</tr>
<tr>
<td>[122]</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>Game-based</td>
<td>Delay</td>
</tr>
<tr>
<td>[123]</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>Lyapunov optimization</td>
<td>Delay</td>
</tr>
<tr>
<td>[126]</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>DQN</td>
<td>Performance</td>
</tr>
<tr>
<td>[127]</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>Many-to-one matching algorithm</td>
<td>Cost</td>
</tr>
<tr>
<td>[129]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>ADMM</td>
<td>Revenue</td>
</tr>
<tr>
<td>[130]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Heuristic-based algorithm</td>
<td>Latency</td>
</tr>
<tr>
<td>[132]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Penalty dual decomposition technique</td>
<td>Delay</td>
</tr>
<tr>
<td>[135]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>DQN, FL</td>
<td>Performance</td>
</tr>
<tr>
<td>[136]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>ADMM</td>
<td>Energy consumption</td>
</tr>
<tr>
<td>[8]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>ADMM</td>
<td>Utility</td>
</tr>
<tr>
<td>[137]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>ADMM</td>
<td>Utility</td>
</tr>
</tbody>
</table>

TABLE V

COMPARISON OF PAPERS FOCUSING ON RESOURCE ALLOCATION. ACRONYMS USED IN THIS TABLE: NON-DOMINATED SORTING GENETIC ALGORITHM (NSGA), DEEP Q-NETWORK (DQN), ALTERNATING DIRECTION METHOD OF MULTIPLIERS (ADMM), FEDERATED LEARNING (FL)
2) **Resource Placement:** In terms of resource placement, a portion of works focus on how to place ENs [146], [147], [154]–[156]. The location and number of edge services have a crucial impact on both the cost of the edge computing network and users’ average latency. The study in [146] presented a cost-aware cloudlet placement scheme for MEC, considering the cost of cloudlet deployment and the average latency of users. A Lagrange-based heuristic algorithm was used to achieve sub-optimal solutions, and a workload allocation scheme was designed to minimize the delay between users and cloudlet considering the mobility of users. The edge server placement has raised concerns on the expenditure of deployment and operation, the current backhaul network capacity, and non-technical placement constraints. In [147], the authors proposed a new framework for edge server placement aiming to reduce the overall costs of deploying and operating edge computing networks. The framework addressed the server placement problem by implementing service placement and optimization strategies.

Notably, there are lots of current research focusing on service placement. On the one hand, some research study decentralized cloud services to the edge [148], [157]–[161]. Nowadays, many data-intensive tasks are computed at the edge. If the data required for the task is not stored at the edge, it needs to be downloaded from the cloud, which may cause additional delay. Therefore, it is valuable to study how to decentralize cloud data to the edge. Jin et al. in [148] proposed an efficient graph-based algorithm for the data placement problem, aiming to maximize the cache hit rate to reduce the task delay. Combining edge computing and cloud computing to place data for scientific workflows to minimize the transmission time across different data centers, the authors in [158] proposed a self-adaptive discrete particle swarm optimization (PSO) algorithm for the data placement problem. The proposed algorithm considered the bandwidth, the number of the edge, and the storage capacity of the edge that affect transmission delay. Similarly, Chen et al. in [160] also explored the data placement problem for scientific workflows, and they proposed the model based on GA and PSO to solve the problem. On the other hand, more works have studied the service or application placement at the edge based on users’ requirements [150], [151], [162]–[168]. The objective functions and constraints in those works are determined by considering various aspects of the edge computing environment, such as the application (or service) architecture, the edge architecture or the edge-cloud architecture, the network condition, and the network topology. In [150], the authors proposed a service placement mechanism based on a logical edge network to meet users’ needs and the resource constraints of ENs. The proposed service placement mechanism aimed to minimize the number of services placed on ENs to optimize the resource utilization of ENs. The work in [151] studied the load distribution and layout of scalable IoT services, including vertical and horizontal, to minimize the possibility of QoS violations due to edge computing resource constraints. Similarly, the study [166] introduced the problem of dynamic edge computing service placement, which was designed to dynamically deploy IoT services on edge resources to meet QoS requirements such as service delay and bandwidth usage. At present, the difficulty and trend of this subject are how to place tasks with data dependencies when the service or application is composed of multiple dependent tasks. Usually, in the dependent category, related works modeled their service or application by Directed Acyclic Graph (DAG) [169]–[173]. The placement purpose of their research is to find a group of tasks for scheduling, by which the execution time of service or application and energy consumption of MD become reduced. Although built on less powerful hardware, edge computing faces similar challenges as cloud computing in effectively managing the hardware resources. Therefore, edge computing also employs virtualization as one of its fundamental technologies. The virtualization technology, no matter in the form of VMs or containers, provides flexible and reliable services for edge computing at a high level. VM placement is a popular research in resource provisioning at the edge, which can be regarded as a process to find the optimal network path to allocate VM. Therefore, the task can be quickly executed, and energy usage can be reduced. Li and Wang [154] proposed the method to find out a VM placement scheme that can reduce the total energy consumption and keep the access delay in a reasonable range. In [174], the authors exploited the prediction of users’ movement. The prediction is used for dynamic VM placement and to find the most suitable communication path according to expected users’ movement. To date, there are several pioneer projects proposed by the industry that aims at building general-purpose edge computing frameworks, including OpenStack [175], Kubernetes [176], and OpenEdge [177]. Applying container techniques to the edge environment is a natural trend because of the facts of rapid construction, instantiation, and initialization of virtualized instances [178]. Morabito [179] evaluated the performance of container-based virtualization on IoT devices on the edge. They conducted more practical experiments on Advanced RISC Machine (ARM)-based IoT end-devices (Raspberry Pi). Performance evaluation on the CPU, memory, disk I/O, and network shows that container-based virtualization can represent an efficient and promising way to enhance the features of edge architectures. In [180], the authors found that inter-container communications, and container management consume significant CPU resources by experiments. Then, a joint task scheduling and containerizing scheme are introduced to tackle this problem. In the past two years, research on resource provisioning based on serverless computing architecture has attracted much attention [152], [153], [181]. Serverless computing is an emerging paradigm for running user-specified functions on resource providers with infinite scalability. Suresh and Gandhi in [152] proposed Fnsched, a novel resource provisioning framework that aims to meet users’ performance requirements while minimizing the cost of SPs. Fnsched implemented the autoscale ability by carefully regulating resource usage on each resource scheduler. Besides, the authors in [153] proposed an MPSC framework for serverless computing that supports multiple edge resource providers. MPSC monitored the performance of serverless providers in real-time and dispatched users’ application tasks to appropriate resources.

A comparison of papers focusing on resource provisioning is presented in Table VI. Since the virtualization technology...

---

A comparison of papers focusing on resource provisioning is presented in Table VI. Since the virtualization technology...
IV. KEY TECHNIQUES AND PERFORMANCE INDICATORS

Advanced scheduling strategies and techniques are indispensable for realizing optimal scheduling of edge computing resources and thus meeting the QoS and QoE requirements of both end-devices and the system. In recent years, many state-of-the-art resource scheduling techniques have emerged. Based on whether a control center is needed to collect global information, resource scheduling can be operated in centralized manner or distributed manner. Generally, centralized methods mainly include convex optimization, approximate algorithm, heuristic algorithm, and machine learning; distributed methods mainly include game theory, matching theory, auction, federated learning (FL), and blockchain, as shown in Fig. 8. In the following, we elaborate on the centralized and distributed resource scheduling methods before summarizing six performance indicators, i.e., latency, energy consumption, cost, utility, profit, and resource utilization.

A. Centralized Methods

1) Convex Optimization: The optimization models developed in the issues of computation offloading, resource allocation, and resource provisioning are typically non-convex or NP-hard problems. A significant portion of studies transform the non-convex problem into a near-convex or convex optimization problem, thus adopting a feasible convex optimization method. Deng et al. in [46] studied the offloading problem under the green and sustainable MEC framework for the IoT system. To minimize the response time, they proposed a DPCOEM algorithm based on the Lyapunov technique and achieve approximately optimal performance. Similarly, some research [87], [182]–[186] also used Lyapunov technique to solve the optimization problem. Lyapunov optimization, as a stochastic optimization approach, can enable online decision-making while preserving sub-optimal performance. The work [187] modeled the problem of resource allocation in MEC as a mixed-integer program. Due to the NP-hardness nature of the formulated problem, the authors proposed a decomposition method to solve it. They decomposed the original problem into two sub-problems, one is the workload assignment and another is the edge node dimensioning. Also, the studies in [188], [189] employed the decomposition method to solve the complicated optimization problem. The authors in [40] investigated the computation offloading problem in the UAV scenario, and the formulated non-convex optimization problem was solved using the Dinkelbath algorithm and successive convex approximation (SCA) technique. Similarly, Liu et al. [190] also used the SCA technique to solve a non-convex optimization problem. The idea of SCA is to iteratively solve a series of convex optimization problems similar to the original non-convex problem, to find a local optimal solution of the original problem. Yang et al. in [191] formulated a non-convex problem for computation offloading and data caching. To solve the problem, they transformed it into a near-convex problem and then designed an algorithm based on ADMM. ADMM is a simple method for solving decomposable convex optimization problems. Using the ADMM algorithm, the original problem can be equivalently decomposed into some solvable sub-problems, which can be solved in parallel. Finally, the solutions of the sub-problems were coordinated to obtain the global solution of the original problem. Besides, the ADMM technique was also utilized in [192], [193].
Summary: The main techniques of convex optimization include the Lyapunov technique, decomposition technique, SCA technique, and ADMM technique. In general, techniques based on convex optimization have the following advantages: a) mature, and widely used; and b) sub-optimal optimization results can be easily obtained. However, the calculations of methods based on these techniques are often complex and challenging to implement in real systems.

2) Approximate Algorithm: In addition to the transformation to traditional convex optimization methods, a large number of studies adopt various approximation algorithms to solve the non-convex and NP-hard problems in resource scheduling. For MEC systems, Badri et al. in [194] built the application placement problem as a multi-stage stochastic programming problem. They adopted a parallel sample averaging approximation (SAA) algorithm to solve this problem and obtained an effective solution. In [195], the computation problem was modeled as an infinite horizon average cost Markov decision process (MDP) process and was approximated to a virtual continuous-time system before a multi-level offloading policy was proposed. The work in [196] studied the edge-cloud placement problem and described it as a multi-objective optimization problem, which was solved by an approximate method using k-means and hybrid quadratic programming. Lu et al. in [197] modeled a multi-user resource allocation problem in edge computing and utilized an approximation algorithm for local search to solve the NP-hard problem. The work in [198] studied the problem of maximizing revenue by placing multiple services in an edge system. The authors first proved that the formulated problem is NP-hard and then proposed a deterministic approximation algorithm to solve it.

Summary: The basic idea of the approximate algorithm is utilizing the existing approximate methods, such as relaxation, bounded, local search, and dynamic planning techniques, to solve the established NP-hard problems. In general, the approximate algorithm has the following advantages: a) simple, flexible, and easy to implement; and b) not difficult to design a local search algorithm for most difficult NP-hard problems. However, the approximation algorithm has some disadvantages: a) easy to fall into a local optimum; and b) the performance of the solution cannot be guaranteed due to randomness.

3) Heuristic Algorithm: Nowadays, one of the most popular ways to solve NP-hard problems is utilizing heuristic algorithms including simple heuristics and meta-heuristics. Using principles similar to bionics, heuristic algorithms abstract some phenomena in nature and animals into algorithms to deal with corresponding problems [199]. In resource scheduling research, most of the current works utilize greedy algorithms while some works utilize local search algorithms. Huang et al. in [161] modeled a multi- replica data placement problem for MEC. They analyzed the complexity of the formulated problem and designed a greedy strategy to solve the problem. Similarly, the works in [116], [200] also employed the greedy idea to solve the NP-hard problem. The study in [155] jointly studied the problem of edge server placement and application allocation, and they proposed a heuristic algorithm based on local search to effectively solve the problem. Likely, the local search heuristic algorithm was also used in [201]. Meta-heuristics in heuristics is widely used in various fields, including genetic algorithm, ant colony algorithm, PSO, simulated annealing, and tabu search. Canali and Lancellotti in [202] designed a heuristic algorithm based on a genetic algorithm for the service placement problem. There are also some works [88], [121], [203]–[206] utilizing the non-dominated sorting genetic algorithm (NSGA) to solve the formulated multi-objective optimization problem. Hu and Li in [205] formulated the request scheduling problem as a mixed-integer nonlinear program. The problem was analyzed as a double decision-making problem, and the authors presented an optimization approach based on NSGA to address the problem. Besides, the authors in [207] proposed a PSO-based heuristic strategy to solve the joint problem of service placement and task provisioning. The study in [208] designed a heuristic algorithm based on tabu search for task scheduling in IoVs. In [209], the authors studied the problem of computation offloading and resource allocation and solved the upper-level optimization problem with an ant colony based heuristic algorithm.

Summary: The research that utilizes heuristic algorithms to solve NP-hard problems in resource scheduling tends to employ greedy-based and genetic-based algorithms. The simple heuristic algorithm is efficient, but easy to fall into a local optimal solution. The meta-heuristic algorithm has too many parameters, which makes it difficult to reuse the calculation results. Also, it is impossible to adjust those parameters quickly and effectively.

4) Machine Learning: In recent years, advanced AI techniques have been applied in various fields due to the development of machine learning, such as deep learning and reinforcement learning techniques. In the research on resource scheduling for edge computing, traditional methods (e.g., convex optimization and approximation algorithms) are usually static solutions to complex optimization problems. They cannot achieve optimal decisions based on dynamic environments. Generally, the interaction with the edge environment during resource scheduling can be modeled as an MDP problem, which can be effectively solved by the reinforcement learning technique. Therefore, many studies utilize reinforcement and deep learning methods for resource scheduling problem in edge computing. In [210], the authors modeled the online offloading problem as an MDP and proposed a deep Q-network (DQN) technique to accommodate dynamic environments and solve the problem. Ning et al. in [211] utilized the DQN technique to design an intelligent scheduling approach for VEC. Similarly, the works in [214]–[216], [217] and [218] respectively studied the computation offloading, resource allocation, and request scheduling problems of IoT users, and all utilized the DQN technique to learn the optimal strategy. Lu et al. in [212] utilized the LSTM network layer and candidate network combined with the actual edge computing environment to improve the DQN algorithm and achieve better performance. The work in [213] studied the computation offloading optimization problem and proved it is NP-hard before proposing an offloading algorithm based
on DQN and FL. Besides, the work in [219] described the offloading decision problem as a multi-label classification problem and utilized a deep supervised learning technique. Chen et al. in [220] proposed a novel prediction-enabled feedback control with reinforcement learning based resource allocation method, which effectively obtain adaptive and efficient resource allocation for cloud-based software services.

Summary: Generally, the machine learning technique used for resource scheduling in edge computing has the following advantages: a) strong parallel processing capability; b) strong distributed storage and learning capability; and c) has the function of associative memory and can fully approximate the complex nonlinear relationship. However, it also has the following disadvantages: a) require a large number of parameters; b) a black-box process, and the learning process cannot be observed, and the output results are difficult to interpret, which will affect the credibility and acceptability of the results; and c) long learning time, and may fall into a local optimal solution or may not even achieve the learning purpose.

B. Distributed Methods

1) Game Theory: Game theory is a powerful framework to analyze the interactions among entities that act for their self-interests with low complexity [221]. In a game, all players are rational and aware that their interests are affected by others and also affect others. All players can change their actions in response to others’ actions to maximize their own interests. Li et al. [222] proposed a game-theoretic scheme to optimize the offloading strategy considering computing resource and bandwidth to minimize the system cost. Liu and Liu [223] formulated a Stackelberg game to model the interactions between ENs and users, where the EN determines the price at which services are provided to maximize its revenue, and users make offloading decisions based on the price to minimize their own costs. Also, Ranadheera et al. [224] developed a distributed mechanism for computation offloading by utilizing a minority game-based method, aiming to guarantee users’ QoE requirement for latency and energy-efficient activation of servers. Similarly, some research [48], [225]–[227] also utilized game theory to analyze and solve the resource scheduling problem in edge computing. Besides, some solutions combine game theory with other techniques. For example, Meng et al. [228] proposed a game-theoretic based resource allocation mechanism to optimally allocate resources for each component task of a mobile application. They combined the mechanism with a reverse-auction based allocation mechanism and a Partial Critical Path (PCP) strategy. Zhan et al. in [229] proposed a computation offloading game framework that does not need information of network bandwidth and preference. To obtain the optimal offloading decision for a maximal utility in terms of processing time and energy consumption, an MDP and a policy gradient based deep reinforcement learning (DRL) are utilized to solve the problem. Zhang [230] proposed a coalitional game-based method to analyze the data offloading from MDs to MEC servers, aiming to improve bandwidth efficiency and user latency, and gain the payoff of MEC servers. To stimulate the offloading, the authors utilized a pricing mechanism to combined with the coalitional game-based method.

Summary: The basic idea of a game theory-based distributed method is to regard each user in the game as a player. The best response decision is made through a collaborative or non-collaborative manner among players to gain their best interests. All those game theory-based methods need to prove the existence of Nash Equilibrium, where a mutually satisfactory solution among users is obtained, and no user is willing to change its decision unilaterally. Generally, the game theory-based method has the following advantages: a) simple, flexible, and easy to implement; and b) practical and rational for the participants. However, it also has the following disadvantages: a) the mutually satisfactory solution may not be the global optimal solution; and b) continuous iteration to achieve the Nash Equilibrium.

2) Matching Theory: The matching theory is a sub-field of economics, which is a promising concept in distributed resource management and scheduling. Besides, the matching theory provides distributed self-organizing solutions to resource scheduling problems with low complexity. In matching theory-based resource scheduling, each agent (such as an EN, a radio resource, or a transmitter node) sorts the others and allocates resources using a preference relation. Generally, a match is defined as: for a given graph $G = (V, E)$, a match of the graph $M$ is a sub-graph of $G$ that consists of a portion of vertexes and edges of the original graph $G$. And there are no common vertex and no adjacent edge in the subgraph. A vertex has at most one edge in a matching graph, and if a vertex has one edge, this vertex is called a matched vertex. Gu et al. [73] studied the problem of how to efficiently assign computing tasks to reduce energy consumption in the edge computing system under the constraints of the computing capacity of both MDs and ENs, wireless channel conditions, and delay. In this regard, this paper utilized a one-to-many matching theory for modeling and analysis, and proposed a heuristic swap-matching based algorithm to solve the task assignment problem. Pham et al. [231] proposed two matching algorithms to solve the computation offloading decision problem and joint resource allocation problem, aiming to minimize the system-wide computation overhead. Similarly, the study in [44], [108], [149], [232] also utilized matching theory-based methods to solve resource scheduling problems in edge computing.

Summary: Matching theory is a strong tool for analyzing the mutually and dynamic beneficial relations between users and SPs [241], [242]. Generally, the matching theory-based method has the following advantages: a) effective in high dynamic networks; and b) extendable, decentralized, and practical for some complex networks. However, since it is generally used to solve binary offloading problems, it is not very appropriate in solving partial offloading problems.

3) Auction: Auction is inherited from economics and is widely used for resource management and scheduling problems. In an auction mechanism framework for resource scheduling, the entities with tasks to be processed act as bidders, and the entities providing task processing service act as
sellers. A trusted entity acts as a third auctioneer to administrate trading and makes online decisions. To understand the auction concept easily, we take the work in [243] as an example. IoT devices first published their computation tasks and the corresponding rewards to the edge computing system. Then, the MDs providing computing services analyzed the rewards they can obtain through computing tasks and submitted their bids to the system. Finally, the system assigns the task to the MD who submitted the highest bids. The auction-based resource scheduling technique can provide a polynomial complexity solution, which has been verified to achieve near-optimal performance. He et al. in [233] considered regarding the resourceful MDs as collaborative nodes to process tasks offloaded from end-devices. And an online auction-based incentive mechanism is proposed to maximize the long-term system welfare. Sun et al. in [45] investigated joint resource allocation and network economics in edge computing. They proposed two double auction schemes with dynamic pricing in MEC to maximize the number of successful trades, one is called breakeven-based double auction (BDA), and another is called dynamic pricing based double auction (DPDA). Li and Cai in [234] integrated time scheduling, resource allocation, and task executor selection for collaborative task offloading, and proposed an online auction mechanism based on primal-dual optimization framework to maximize the social welfare. Also, the work in [235] proposed a reverse auction theory-based method to solve the 0-1 nonlinear integer programming optimization problem to decide the offloading target channel. Similarly, the research in [119], [244] also utilized the auction-based method to solve resource scheduling problem in edge computing.

**Summary:** Like the game theory-based method, in an auction-based resource scheduling framework, both SPs and users try to maximize their own welfare. Generally, the matching theory-based method has the following advantages: a) economic efficiency to achieve a trade-off between requests and services; and b) practical in real scenarios. However, it also has the following drawbacks: a) the solution may not be the global optimal solution; and b) extra third trusted party for auction management may induce extra overhead.

4) **Federated Learning:** FL, also known as collaborative learning, is a machine learning technique that can train resource scheduling algorithm on multiple distributed edge devices or servers that do not exchange local data samples [245]. FL is a distributed machine learning algorithm, which not only takes the advantages of machine learning in solving dynamic resource scheduling problems, but also develops and improves it. In this regard, Ren et al. in [236] studied the computation offloading problem for IoT devices in an energy harvesting scenario. To jointly allocate communication and computing resources during the offloading process, DRL agents are deployed in IoT devices to guide them to make offloading decisions. Meanwhile, to make the DRL-based algorithm feasible and reduce the transmission overhead between IoT devices and servers, the FL method is adopted to train DRL agents in a distributed manner. Also, to jointly allocate communication, computing, and storage resources in edge computing, the authors in [135] integrated the DRL method and FL method in edge computing and proposed an In-Edge-AI framework, where the parameters of the training model are exchanged between end-devices and edge node to better optimize the resource scheduling model. Besides, Qian et al. in [237] combined the FL method with a centralized greedy algorithm to address the problem of service placement with privacy-awareness in the edge computing system.

**Summary:** Compared with the traditional centralized machine learning algorithm, FL has the following advantages: a) since the training process is carried out on distributed devices, there is no need to upload local data to the dedicated server for centralized training, which can protect the user privacy and reduce the data transmission burden of wireless channels; b) users only upload the parameters of their own training models, and the synthesized parameters from multiple devices are fed back to users, which can effectively reduce the individual training time. However, it also has the following disadvantages: a) involves in multiple devices; and b) is vulnerable to malicious attacks. The FL method for resource scheduling in edge computing is a new method, and we look forward to more works in the future.

5) **Blockchain:** Blockchain technology, as an emerging decentralized security system, has attracted more and more attention due to its unique functions such as decentralization, non-tampering, irreversible and traceable, and has been applied in many applications, such as bitcoin, smart grid, and IoT [235], [246]. The introduction of blockchain technology into edge computing can ensure the integrity of resource transaction data and the SP’s profits. There are several works considering integrating the blockchain technology into edge computing [238]–[240], [247]. To manage edge computing resources effectively, the work in [238] introduced a novel concept of edge computing for mobile blockchain and presented a prototype for IoT blockchain mining tasks offloading. Xu et al. in [239] proposed BCD, a blockchain-based computation offloading method in edge computing. The proposed method can address the unequal resource distribution problem and ensure QoS requirements of users with an offloading strategy that preserves data integrity and balance. Also, to ensure the integrity of resource transaction data and SPs’ profits, Xiao et al. in [240] proposed an emerging IoT architecture, name EdgeABC, where the computation offloading algorithm is implemented on the blockchain in the form of smart contracts.

**Summary:** The blockchain-based method has the following advantages: a) can maintain data security; and b) can maintain data integrity. However, it also has the following disadvantages: a) has relatively high latency; and b) involves in multiple devices. The blockchain-based resource scheduling method in edge computing is also a new method, we expect more future works dedicated to this direction.

From the above analysis, since centralized methods need to collect global information from users, it can obtain a better optimal solution and incur more overhead than distributed methods. Differently, distributed methods are more simple, flexible, easy-implement, and adaptive to a dynamic environment than centralized methods. We summarize centralized and distributed methods in Tables VII and VIII, respectively.
TABLE VII
COMPARISON OF PAPERS USING CENTRALIZED METHODS. ACRONYMS USED IN THIS TABLE: MARKOV DECISION PROCESS (MDP), SUCCESSIVE CONVEX APPROXIMATION (SCA), ALTERNATING DIRECTION METHOD OF MULTIPLIERS (ADMM), NON-DOMINATED SORTING GENETIC ALGORITHM (NSGA), DEEP Q-NETWORK (DQN), QUALITY OF SERVICE (QoS), QUALITY OF EXPERIENCE (QoE), LONG SHORT-TERM MEMORY (LSTM), FEDERATED LEARNING (FL)

<table>
<thead>
<tr>
<th>Tech.</th>
<th>Paper</th>
<th>Objective</th>
<th>Online</th>
<th>Method</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex optimization</td>
<td>[46]</td>
<td>Response time</td>
<td>×</td>
<td>a) Use Lyapunov technique to decompose the formulated problem to be a convex optimization; b) Proposed a DPCOEM algorithm to solve the problem.</td>
<td>a) Mature and widely used; b) Near-optimal results can be easily obtained.</td>
<td>a) High complexity; b) Poor practicality.</td>
</tr>
<tr>
<td></td>
<td>[187]</td>
<td>Cost</td>
<td>×</td>
<td>a) Divide the formulated problem into two sub-problems; b) Propose a trade-off approach to solve it.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[40]</td>
<td>Energy efficiency</td>
<td>×</td>
<td>a) Decompose the problem into sub-problems; b) Use the Dinkelbach algorithm and SCA technique to solve it</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[191]</td>
<td>Execution delay</td>
<td>×</td>
<td>a) Use McCormick envelopes to transformed the problem into a near-convex one; b) Designed an algorithm based on ADMM to achieve near optimal results.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approximate algorithm</td>
<td>[194]</td>
<td>QoS</td>
<td>×</td>
<td>a) Use a sample averaging approximation algorithm to solve multi-stage stochastic programs; b) Design a fast parallel greedy algorithm to solve application placement.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[196]</td>
<td>Service delay</td>
<td>×</td>
<td>a) Prove the formulated problem is NP-hard; b) Propose an approximate approach with k-means and hybrid quadratic programming.</td>
<td>a) Simple, flexible and easy to implement; b) Easy to design a local search algorithm.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[197]</td>
<td>Cost</td>
<td>✓</td>
<td>a) From a simple case to a complicated case; b) Prove the formulated problem is NP-hard; c) Propose an approximation algorithm for local search.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[198]</td>
<td>Revenue</td>
<td>✓</td>
<td>a) Prove the formulated problem is NP-hard; Propose a deterministic approximation algorithm to solve it.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heuristic algorithm</td>
<td>[161]</td>
<td>Latency</td>
<td>×</td>
<td>a) Prove the problem is NP-hard; b) design a greedy-based heuristic algorithm to address it.</td>
<td>a) Easy to fall into the local optimum; b) The performance of the solution cannot be guaranteed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[155]</td>
<td>Service cost</td>
<td>×</td>
<td>a) Prove the formulated problem is NP-hard; b) Propose SPAC based on local search.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[202]</td>
<td>Latency</td>
<td>×</td>
<td>a) Prove the formulated problem is NP-hard; Propose a scalable heuristic approach based on genetic algorithm.</td>
<td>a) Efficient; b) Obtain the optimal solution quickly</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[205]</td>
<td>Latency</td>
<td>×</td>
<td>a) Analyze the problem as a double decision-making problem; b) Propose an heuristic approach based on NSGA.</td>
<td>a) Easy to fall into the local optimum solution; b) Too many parameters.</td>
<td></td>
</tr>
<tr>
<td>Machine Learning</td>
<td>[210]</td>
<td>Performance</td>
<td>✓</td>
<td>a) Formulate the offloading problem as an MDP; b) design a DQN-based offloading policy.</td>
<td>a) Strong parallel processing capability; b) Strong distributed storage and learning capabilities; c) Approximate the complex nonlinear relationship.</td>
<td>a) Require a large number of parameters; b) A black-box process; c) Long learning time.</td>
</tr>
<tr>
<td></td>
<td>[211]</td>
<td>QoE</td>
<td>✓</td>
<td>a) Divide the original problem into two sub-problems; b) Develop a two-side matching scheme and a DQN approach to schedule requests.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[212]</td>
<td>Performance</td>
<td>✓</td>
<td>a) Propose a DQN algorithm to solve the offloading problem; b) use LSTM network layer and candidate network to improve DQN algorithm.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[213]</td>
<td>Utility</td>
<td>✓</td>
<td>a) Prove the formulated problem is NP-hard; b) Design an offloading method based on DQN and FL.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C. Performance Indicators

1) Latency: From the objectives designed in current research (Table IV–Table VIII), we find that latency is a key performance indicator that affects users’ QoE. For delay-sensitive applications, designing a resource scheduling algorithm to reduce latency is one of the main focuses. Since the computing, communication, and storage resources in the edge system are limited, if multiple delay-sensitive task requests are sent to the edge simultaneously, not only the latency requirements should be considered but also the constraints of resource capacity and energy consumption should be weighed, which would form a complex optimization problem. Generally, the latency of a task in resource scheduling consists of: a) local computing time; b) transmission time for task offloading; c) processing time at the edge or cloud; and d) transmission time for result return. The idea of current research is generally establishing a delay model for specific application scenarios, and formulating an optimization problem by considering various constraints to reduce latency, before solving it by different algorithms.

2) Energy Consumption: Energy consumption is an important performance indicator for users’ QoE in edge computing system, especially for small smart devices. The energy consumption in the research of resource scheduling in edge computing mainly consists of: a) the energy consumption for local computing; b) the energy consumption for offloading; c) the energy consumption for processing tasks at the edge or cloud; and d) the energy consumption for transmitting result back. Many works just aim to reduce energy consumption [201], [248], [249] while some works aim to reduce latency and energy consumption simultaneously [51], [107], [112], [131], [250]. Besides,
TABLE VIII
Comparison of Papers Using Distributed Methods. Acronyms Used in This Table: Markov Decision Process (MDP), Deep Reinforcement Learning (DRL), Non-Dominated Sorting Genetic Algorithm (NSGA), Vehicular Edge Computing (VEC), Mobile Device (MD), Edge Node (EN), Federated Learning (FL)

<table>
<thead>
<tr>
<th>Tech.</th>
<th>Paper</th>
<th>Objective</th>
<th>Online</th>
<th>Method</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Game Theory</td>
<td>[222] Cost</td>
<td>×</td>
<td>a) The formulated problem is decoupled into resource allocation and offloading decision-making problems; b) The offloading decisions are obtained via potential game; c) The resource allocation is achieved by using the Lagrange multiplier.</td>
<td>a) Simple, flexible and easy to implement; b) Practical and rational strategy for the participants.</td>
<td>a) The mutually satisfactory solution may not be the global optimal solution; b) Continuous iteration to achieve the Nash Equilibrium.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[223] Revenue, cost</td>
<td>×</td>
<td>Depending on the edge node's knowledge of the network information, developed the uniform and differentiated pricing algorithms.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[224] Energy efficiency</td>
<td>✓</td>
<td>A distributed learning algorithm to solve server mode selection problem</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[229] Utility</td>
<td>✓</td>
<td>a) Formulate the problem as a partially observable Markov Decision Process (MDP); b) Solve it by a policy gradient based approach.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Matching Theory</td>
<td>[231] Overhead</td>
<td>×</td>
<td>a) Users make the offloading decisions; b) Approximate the inter-cell interference and find the transmit power of offloading users using a bisection method.</td>
<td>a) Effective in high dynamic networks; b) Extendable, decentralized, and practical solutions for some complex networks.</td>
<td>a) Generally used to solve binary offloading problem; b) Ineffective in solving partial offloading problem.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[232] Delay</td>
<td>×</td>
<td>a) Formulate the task assignment problem in VEC as a matching game; b) Propose two methods, one is one-to-many matching method and another is a heuristic swap-matching method.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[44] Throughput</td>
<td>✓</td>
<td>Propose a learning-based channel selection framework by leveraging the combined power of machine learning, Lyapunov optimization, and matching theory.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auction</td>
<td>[233] Welfare</td>
<td>✓</td>
<td>a) Propose a VCG-based offline optimal auction Mechanism; b) Propose a Myerson Theorem-based allocation rule of online truthful auction.</td>
<td>a) Economic efficiency to achieve a trade-off between requests and services; b) Practical in real scenarios.</td>
<td>a) The solution may not be the global optimal solution; b) Extra overhead will be induced since a third trusted party is needed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[45] Successful trades</td>
<td>×</td>
<td>a) Propose a break-even-based double auction (BDA); b) Propose a more efficient dynamic pricing based double auction (DPA) solution.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[234] Welfare</td>
<td>✓</td>
<td>a) Proposed a primal-dual framework based online auction. b) Schedule transmission and computing times, and optimally allocate communication and computing resources.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[235] Energy consumption</td>
<td>×</td>
<td>a) Determine the MD user classification and priority; b) Proposed a reverse auction-based offloading algorithm.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Federated Learning</td>
<td>[236] Utility</td>
<td>×</td>
<td>a) Multiple DRL agents are deployed on multiple ENs to indicate the decisions of the IoT devices; b) FL is used to train DRL agents in a distributed fashion.</td>
<td>a) Privacy-protected; b) Reduce the burden of wireless channel; c) Low overhead of learning.</td>
<td>a) Involve in multiple devices; b) Vulnerable to malicious attacks.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[135] Utility</td>
<td>✓</td>
<td>a) Integrate the DRL and FL methods with edge computing system; b) Exchange the training model parameters among end-devices and servers in a collaborative way.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[237] Privacy, service demands</td>
<td>×</td>
<td>a) Model the problem of whether service is placed on edge node or not as a 0-1 problem; b) Propose a hybrid algorithm combining a distributed FL method and a centralized greedy algorithm.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blockchain</td>
<td>[238] Profit</td>
<td>×</td>
<td>a) A prototype of an edge computing system for mobile blockchain; b) A pricing schemes.</td>
<td>a) Maintain data security; b) Maintain data integrity</td>
<td>a) Relatively high latency; b) Involve in multiple devices.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[239] Latency</td>
<td>×</td>
<td>a) blockchain-based framework is designed degrade the data loss possibility; b) NSGA-III is leveraged to acquire the balanced offloading strategies;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[240] Profit</td>
<td>×</td>
<td>a) subtask-virtual machine mapping strategy; b) stack cache supplement mechanism.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

there are also some works considering end-devices have the function of energy harvesting and wireless charging during the energy consumption minimization [72], [79], [111], [186].

3) **Cost:** Research on minimizing the cost of the edge computing system as a performance indicator is generally a comprehensive performance indicator established under satisfying user service quality. As described in Section III-A, when the task is offloaded, its costs include the energy cost (for transmission and processing tasks), the cost for using communication channels for transmission, and the cost for processing tasks at the edge. The current research generally seeks the best solution by establishing different cost models with the objective of minimizing the cost [71], [214], [251].

4) **Utility:** The concept of utility in edge computing refers to the satisfaction users obtain under a certain resource scheduling scheme. And the utility is generally represented by the utility function. According to different objectives, the utility function is represented and mathematically transformed by
different service quality parameters, such as data transmission rate, delay, energy consumption, and cost. The mathematical transformation mainly includes reciprocal, logarithm, and weighted summation. Finally, effective optimization algorithms are designed to maximize the utility [91], [108], [249], [252], [253].

5) Profit: The profit is generally measured from the perspective of edge SPs when deploying, allocating, and scheduling edge resources for users. The obtained profit is calculated by subtracting the SPs’ operating costs from users’ payment. Under the condition of satisfying the users’ QoS, a profit maximization problem is generally developed before some marvelous solutions (such as game theory, matching theory, and auction) is proposed [145], [254]. Similarly to profit maximization problem, some works also aim to maximize the welfare of society in edge computing system [233]–[235].

6) Resource Utilization: Resource utilization is also measured by edge resource providers. Since the resources in edge are limited compared to that in cloud, the utilization of edge resources becomes particularly important with the increasing users. A proper resource scheduling strategy can make full advantage of edge resources and meet users’ requirements simultaneously. Existing works typically aim to maximize resource utilization, which is defined as the ratio of the resource usage volume to the total resource volume [217], [255]–[257].

V. RESOURCE SCHEDULING IN APPLICATIONS CONTEXT

New applications are the main driving force for edge computing. Edge computing involves optimal resource scheduling in many application scenarios due to users’ stringent requirements for latency, energy consumption, cost, privacy, etc. In this section, we introduce several typical application scenarios involved in the research on resource scheduling in edge computing. When we were analyzing references, we recorded the applications involved in each paper. Through statistics, we have summarized several more researched and more common applications, which serve as the typical applications of this survey, including UAV, CAV, video service, smart city, smart health, smart manufacturing, and smart home, as shown in Fig. 9.

A. UAV

UAVs, especially low-cost quad-rotor aircraft, are experiencing explosive growth and have been widely used in civil and military fields, such as traffic monitoring, public safety, disaster detection, search, and rescue. And the research on resource scheduling in the field of UAVs can be divided into two directions:

1) UAVs as Users: In some computing-intensive applications, the UAVs are unable to meet the task requirements due to the limited resources. In this case, the resources at the edge of the wireless network, such as cellular BSs, can provide cloud-like computing services to assist UAVs to complete the task [258], [259]. Cao et al. in [258] studied how to offload the latency-sensitive tasks of UAVs to the ground BSs, subject to the speed constraint of UAVs. Similarly, the authors in [259] studied the offloading problem based on two-tier UAVs, aiming to minimize the latency of tasks and the system cost.
TABLE IX

<table>
<thead>
<tr>
<th>Paper</th>
<th>Research issue</th>
<th>Edge</th>
<th>What’s to be scheduled</th>
<th>Key points</th>
</tr>
</thead>
<tbody>
<tr>
<td>[258]</td>
<td>Computation offloading</td>
<td>BSs</td>
<td>Tasks from UAVs</td>
<td>Minimize the response time; Optimizing the trajectory of UAVs; the constraints: the speed of UAVs and the computation capacity of BSs</td>
</tr>
<tr>
<td>[259]</td>
<td>Computation offloading</td>
<td>BSs</td>
<td>Tasks from MUs</td>
<td>Minimize latency and cost; Stackelberg game</td>
</tr>
<tr>
<td>[260]</td>
<td>Joint deployment and task scheduling</td>
<td>UAVs</td>
<td>Tasks from MUs</td>
<td>Minimize system energy consumption; a two-layer optimization method</td>
</tr>
<tr>
<td>[261]</td>
<td>Joint Computation offloading and trajectory scheduling</td>
<td>UAVs</td>
<td>Tasks from MUs</td>
<td>Maximize computation efficiency; the constraints: user association, computing and spectrum resources; non-convex problem</td>
</tr>
<tr>
<td>[107]</td>
<td>Joint task offloading and resource placement</td>
<td>UAVs and ESs</td>
<td>Tasks from MUs</td>
<td>Maximize service delay; maximize the energy efficiency; non-convex problem</td>
</tr>
<tr>
<td>[250]</td>
<td>Joint UAV deployment and computation offloading</td>
<td>UAVs and ESs</td>
<td>Tasks from MUs</td>
<td>Maximize task delay and energy consumption</td>
</tr>
</tbody>
</table>

2) UAVs as Edge Resources: Due to the convenient mobility, UAVs can be regarded as mobile edge resources or cooperate with traditional edge servers on the ground to improve their connectivity, which can provide high-quality services for users [107], [250], [260]–[263]. In [260], multiple UAVs are regarded as flying edge nodes for MUs. The authors presented ToDeTaS, a two-layer optimization method, to jointly solve the deployment and task scheduling problem, aiming to minimize the system energy consumption. Likely, Zhang et al. in [261] formulated a computation efficiency maximization problem in a UAV-assisted MEC system. Yu et al. in [107] proposed a UAV-enabled MEC system to provide the computing service to the IoT devices, which cannot access any service due to the sparse distribution of the existing ENs. They studied the resource allocation problem to minimize the service delay of IoT devices. Similarly, in [250], under the UAV-aided MEC architecture, the authors studied the task offloading problem and adopted the agent to conduct an offloading plan based on the perceived information of users, UAV, and edge nodes.

We summarize the studies on UAVs mentioned above in Table IX.

B. CAV

With the development of AI, computer vision, depth perception and sensing technologies, vehicles have gradually evolved from traditional travel tools into CAVs with intelligent and interconnected computing systems. According to Intel, 4TB of raw data would be generated from a CAV in one day, which poses a great challenge on processing capacity of CAVs to support various low-latency and computation-intensive applications. Therefore, the research on computation offloading from vehicles to edge or cloud has attracted much attention. Also, considering the enhancement of the computing, communication, and storage capabilities of vehicles and the widespread distribution, vehicles can also be regarded as edge resources to provide users with flexible computing services. Accordingly, the research on resource scheduling in edge computing under the CAV environment includes two directions:

1) Vehicle as Users: In this case, the focus is to schedule the tasks generated by vehicles to the edge (e.g., RSU) [90], [208], [249], [253], [264]–[268]. Li et al. in [253] considered the vehicular edge computing framework where the computation tasks of autonomous vehicles can be scheduled to RSUs. They investigated the task offloading problem based on the time-varying channel characteristics to maximize the system utility. Likely, by offloading vehicles’ tasks to RSUs, the work in [264] took load balancing into account and used FiWi technology to manage network due to the dynamic vehicular network. Then, the authors proposed a soft-defined network (SDN) based offloading scheme aiming to minimize the task delay. Zhou et al. in [265] studied the energy-efficient offloading problem and presented a distribution method based on consensus ADMM. The work in [267] developed a multi-objective optimization problem for computation offloading in an IoV edge system to reduce energy consumption and delay simultaneously. And the authors adopted a non-dominated sorting genetic algorithm to solve the problem. Moreover, the work in [268] formulated a computation offloading problem as a distributed offloading decision-making game, in which each vehicle as a player makes its best response decision to minimize its joint cost (including latency and offloading cost).

2) Vehicle as SPs: In this case, vehicles can be the supplement to the edge, providing computing services for MUs [36], [269]–[271]. Utilizing the idle resources of parked vehicles (PVs), the authors in [269] studied how to schedule the tasks generated by MUs that can be partitioned into subtasks to PVs, aiming to maximize the social welfare. Besides, Huang et al. in [36] regarded PVs as available edge resources that can collaborate with the existing edge servers to provide computing services for MUs. They proposed an interactive protocol for service provisioning considered the security and privacy requirements of users. Similarly, in [270], collaborated with edge servers, PVs are employed to execute tasks of MUs with delay constraints. The authors proposed a distributed approach based on the Stackelberg game to solve the task assignment problem. Particularly, AVE was presented in [271] as a job scheduling framework, where autonomous vehicles collaborate to provide computing services for each other.
We summarize the studies on UAVs mentioned above in Table X.

### C. Video Service

The video generated by smart devices has promoted the development of various applications, such as traffic control, autonomous driving, public surveillance and security, and AR/VR. Due to the limited storage and computing capabilities of smart devices, it may be inefficient to process the computation-intensive and bandwidth-hungry videos locally. Scheduling video service to the edge to process is a feasible method to meet the low-latency requirement.

In [272], VideoEdge was proposed to optimize the placement of computer vision components, where two challenges were addressed including exponentially large search space caused by multiple resource providers and merging conflicts. Yi et al. in [273] presented LAVEA, a video analytics edge computing platform. They formulated the task selection and prioritized for offloading as an optimization problem. LAVEA can provide low-latency computation offloading service based on serverless architecture. For the AR applications in video services, Ali et al. in [274] proposed a resource allocation scheme, which involved both communication and computing resources. They leveraged the inherently collaborative nature of AR applications and solved the energy expenditure minimization problem with low-latency constraint by the successive convex approximation algorithm. Further, Liu and Zhang in [275] considered the reliability of AR task offloading problem, where the components of an AR task was modeled as a directed acyclic graph with dependencies. To minimize the failure probability of AR service, an integer PSO-based algorithm was proposed.

We summarize the studies on video services mentioned above in Table XI.

---

**Table X**

<table>
<thead>
<tr>
<th>Paper</th>
<th>Research issue</th>
<th>Edge</th>
<th>What’s to be scheduled</th>
<th>Key points</th>
</tr>
</thead>
<tbody>
<tr>
<td>[253]</td>
<td>Computation offloading</td>
<td>RSUs</td>
<td>Tasks from vehicles</td>
<td>Maximize the system utility; time-varying channel; the linearization based branch and bound algorithm</td>
</tr>
<tr>
<td>[264]</td>
<td>Computation offloading</td>
<td>RSUs</td>
<td>Tasks from vehicles</td>
<td>Minimize the task delay/load balancing; SDN-based scheme</td>
</tr>
<tr>
<td>[265]</td>
<td>Workload offloading</td>
<td>UAVs</td>
<td>Tasks from vehicles</td>
<td>Maximize the energy efficiency; a low-complexity distributed method based on ADMM</td>
</tr>
<tr>
<td>[267]</td>
<td>Computation offloading</td>
<td>RSUs</td>
<td>Tasks from vehicles</td>
<td>Multi-objective: reduce energy consumption and time delay while keep load balancing; non-dominated sorting genetic algorithm</td>
</tr>
<tr>
<td>[268]</td>
<td>Computation offloading</td>
<td>RSUs</td>
<td>Tasks from vehicles</td>
<td>Distributed offloading decision-making game; self-learning based distributed computation offloading</td>
</tr>
<tr>
<td>[269]</td>
<td>Task offloading and container placement</td>
<td>PVs</td>
<td>Tasks from MUs</td>
<td>Maximize the social welfare; convex optimization methods</td>
</tr>
<tr>
<td>[36]</td>
<td>Service provisioning</td>
<td>PVs and ESs</td>
<td>Tasks from MUs</td>
<td>Maximize the cost of users; an interactive protocol; security and privacy constraints; Stackelberg game approach</td>
</tr>
<tr>
<td>[270]</td>
<td>Task offloading</td>
<td>PVs and ESs</td>
<td>Tasks from MUs and vehicles</td>
<td>Maximize the overall cost; Stackelberg game approach</td>
</tr>
<tr>
<td>[271]</td>
<td>Task offloading</td>
<td>Vehicles</td>
<td>Tasks from vehicles</td>
<td>Maximize the system utility; vehicle-to-vehicle communication; ant colony optimization</td>
</tr>
</tbody>
</table>

**Table XI**

<table>
<thead>
<tr>
<th>Paper</th>
<th>Things</th>
<th>Edge</th>
<th>What’s to be scheduled</th>
<th>Key points</th>
</tr>
</thead>
<tbody>
<tr>
<td>[272]</td>
<td>IoT Cameras</td>
<td>Private clusters and public clouds</td>
<td>Components of computer vision</td>
<td>Maximize the average query accuracy; trade-off between multiple resources and accuracy; the constraints: large search space and merging conflicts</td>
</tr>
<tr>
<td>[273]</td>
<td>Smartphones, security/dash cameras</td>
<td>Container-based ESs</td>
<td>Components of videos</td>
<td>Minimize response time; inter-edge collaboration</td>
</tr>
<tr>
<td>[274]</td>
<td>Smartphones</td>
<td>ESs</td>
<td>Components of a AR application</td>
<td>Minimize the energy expenditure and latency; component-based model of an AR application; successive convex approximation algorithm;</td>
</tr>
<tr>
<td>[275]</td>
<td>MDs</td>
<td>ESs</td>
<td>AR Tasks</td>
<td>Minimize the failure probability; the reliability and latency requirement; the dependency of sub-tasks; PSO-based algorithm</td>
</tr>
</tbody>
</table>
D. Smart City
In 2016, Alibaba put forward the concept of “smart city”, where multiple urban data are used to manage the city better. To manage and process the smart city data characterized by diversity and heterogeneity and involved the privacy and security of residents, some studies focus on designing edge collaborative processing systems [276]–[278]. Also, some works on the optimal placement of edge resources provide convenient and fast computing services for emerging applications in smart cities [150], [202], [257], [279]. For large-scale smart cities, the authors in [150] presented the logical edge network formed in a tree topology to place edge service in a resource-effective way. Based on the logical edge network, they also designed a service placement scheme meeting the service demands of IoT devices as well as the resource capacity of edge servers. To process the quantities of services produced by IoT devices in smart cities, Xu et al. in [279] proposed TSP as a trust-oriented IoT service placement scheme to tackle the improvement of resource usage, load balance and energy consumption while protecting the privacy of IoT devices. Similarly, to deal with data streams generated from sensors deployed in smart cities, Canali and Lancellotti in [202] also studied the service problem and proposed a scalable heuristic-based genetic algorithm.

E. Smart Health
The development of cloud computing, wireless broadband communication, BAN and wearable medical devices enhances mobile medical services and improves medical standards and medical conditions. However, as medical data grows exponentially, the cost of operating and maintaining the medical system is increasing. To alleviate this situation, deploying edge resources to process medical data at the edge has attracted much attention [251], [285], [286]. Moreover, the establishment of edge-assisted medical systems can save costs for healthcare service providers [280], [287], [288]. Alam et al. [251] proposed an edge-of-things (EoT) computation framework for healthcare service provisioning, where an EoT is a bridge between service providers and healthcare consumers. The authors proposed a portfolio optimization approach for cost-effective service provisioning and used an ADMM method for healthcare data offloading. The security and privacy of healthcare data in smart health is very important. In [280], a security provisioning model named AZSPM, was proposed for medical devices in edge computing. AZSPM can build trust among medical devices with zero knowledge. For the wearable smart devices for physical monitoring, the work in [286] proposed an edge computing-based deep learning network system for physical monitoring by using multimedia technology with agile learning for real-time data processing, which improved the multiple performance metrics effectively.

F. Smart Manufacturing
Smart manufacturing refers to the realization of intelligent industrial operations through AI and big data technology. In smart manufacturing, the industrial devices need real-time control based on the generated data characterized with security and privacy. And the introduction of AI technology into IIoT requires powerful computing capabilities to complete advanced fault prediction, demand forecasting and other big data processing tasks. Therefore, applying edge computing in smart manufacturing has become the direction of industry development, which can improve system performance, ensure data security and privacy, and reduce the cost of operation [281], [282], [289], [290]. Chen et al. [289] presented an edge computing architecture for IoT-based manufacturing, where edge computing acted as edge equipment, information fusion, network communication and cooperative mechanism with traditional computing. Job shop scheduling (JSP) problems are complex in smart manufacturing. In [281], Lin et al. proposed an edge computing framework for smart manufacturing, which adjusted DQN to solve JSP problems. The work in [282] designed an AI-enhanced offloading framework that combined the edge and cloud computing to maximize the service accuracy in IIoT. The authors introduced edge intelligence to smart manufacturing for the sake of many advantages it can bring, including personalization, responsiveness and privacy.

G. Smart Home
The development and enrichment of smart devices have made the system of smart homes reaches commercial maturity. Smart homes use lots of IoT devices (such as various sensors) to control and monitor the living environment in real-time. However, the ever-increasing number of smart devices, the multiple applications with low latency requirements, the big data generated by smart devices, and the extremely private home data, make it a trend to apply edge computing instead of cloud computing to smart homes. There are many works focusing on edge resource scheduling towards the smart home environment [218], [283], [291]. EdgeOSH, a home operating system, was proposed in [291] to provide functions of the program interface and data management. In [283], HomePad was presented for home environments, and it allows IoT applications to execute at the edge. For users’ privacy, HomePad was designed to enable users to determine how applications access and process sensitive data generated by smart devices. Besides, Wang et al. in [284] studied the resource management of the healthcare system in smart homes under the edge-cloud architecture, and presented a task scheduling scheme named HealthEdge, which can process different tasks based on priorities aiming to reduce the latency.

The studies on smart city, smart health, smart manufacturing and smart home are called the study on smart “things” in our survey. And we summarize the studies on smart “things” mentioned above in Table XII. Notably, the application scenarios for smart “things” are deeply dependent on the development of IoT. We believe that the research on each application scenario will become more and more mature thanks to the explosive growth of edge computing in the field of IoT.
VI. CHALLENGES AND RESEARCH DIRECTIONS

Despite the fact that the research on resource scheduling in edge computing has accumulated a lot of results, there are still many key issues that have not been well explored. This section discusses several open research challenges followed by future research directions.

A. Model and Architecture

1) Computation and Communication Model: To efficiently schedule edge resources to accomplish task processing, a computation model should be first established to reflect the relationship between task data size and the amount of computing capacity it requires. In most existing works, it always utilizes a processing density (in CPU cycles/bit) to denote this kind of relationship; thus that the amount of computing capacity a task requires is equal to the product of task data size and processing density [10], [64]. Obviously, it is a linear representation. However, since different types of tasks have different processing densities, this kind of one-size-fits-all representation approach may not be suitable for various application tasks in edge computing. Therefore, more flexible computation models are worthy of further study. Besides, to better process application tasks, utilizing communication resources to offload part or all of the tasks to ENs is trending. During this process, the data transmission rate is a key concern for communication resource scheduling. Current representations of data transmission rate are mostly based on the Shannon-Hartley theorem, which tells a theoretical tightest upper bound on the data transmission rate over a communication channel of a specified bandwidth in the presence of noise. However, in the practical scenario of edge computing, end-devices and ENs are always positioned in a complicated environment with extremely poor channel conditions, such as high mobility, shield, and interference [253]. The actual data transmission rate can not achieve the theoretical value. Therefore, it is necessary to develop a more practical communication model based on field tests or considering different application scenarios.

2) Computation Migration: Since task processing always involves cooperation among multiple ENs or end-devices, few studies focus on computation migration. Generally, to accomplish the computation migration, there are mainly six steps: migration environment sensing, task division, migration decision, task uploading, task execution, result return. Among them, task division and migration decision are the two most critical steps. However, in most existing works that considered computation migration in resource scheduling, only the migration decision step is considered, and other steps are ignored [98], [99]. Computation migration is more like a kind of concept of collaborative computing in current studies. Future research can focus more on the implement of computation migration considering the entire process.

3) Task Partitioning and Integration: Computation offloading has attracted much attention in resource scheduling in edge computing. A task can be divided into two parts, one part computed locally and the other part offloaded to ENs or other nodes for processing. It is assumed that the offloaded part of a task is denoted by an offloaded ratio in most existing works [107], [108]. The resource scheduling process is to determine an optimal offloaded ratio and other optimization variables. Once the optimal offloaded ratio is obtained, this part of the task is directly offloaded [17]. However, for a certain task, the divisible part may not be equal to the optimal offloaded part based on the optimization solution. Therefore, future research should step further on exploring the nature of tasks during task partitioning for computation offloading. After the task is partitioned and processed by different nodes, it is necessary to integrate the dispersal results. Another concern may arise during this process: whether the integrated results are the same as those of none-partitioning processing? This concern leads to a future study on how to integrate...
the processing results from different nodes without losing the original information of the task.

4) Green Energy: To achieve energy saving and maintain longer battery life of IoT devices, it is a trend to utilize renewable green resources light and wind to strengthen energy support, which can significantly reduce carbon emissions and environmental pollution. There are many studies on energy-harvesting or wireless-charging enabled edge computing [79], [111]. The introduction of extra energy supplement makes resource scheduling more complex since not only the energy consumption model during task transmission and task processing should be considered, but also the harvested energy. Although marvelous solutions are proposed in existing works, most of them consider the extra energy can be harvested continuously [72], [186]. However, in practice, the energy harvesting process may be unstable, which poses a significant challenge in designing an efficient resource scheduling strategy. Therefore, future research should focus more on the energy harvesting process.

5) Heterogeneous Architecture: The architecture of edge computing generally includes things layer, edge layer, and cloud layer. Most of the existing research on resource scheduling are under the thing-edge-cloud architecture. It is predicted that the integration of multidimensional networks such as space, air, and ground to form the space-air-ground integrated network (SAGIN) is the future trend to support the ever-increasing IoT applications [292]. Under such a space-air-ground heterogeneous architecture, the SAGIN incorporated with edge computing can provide a myriad of services and applications, such as edge caching, computation offloading and cloud services [293]. However, heterogeneous nodes (end-devices, edge servers, CAVs, UAVs, and satellites) and the heterogeneous resources of those nodes make the resource management and scheduling complicated. Besides, heterogeneous nodes are subject to strong spatio-temporal constraints [294], which make the management and scheduling of heterogeneous resources more challenging. Therefore, it is necessary to develop an efficient resource scheduling and management technology that can simultaneously orchestrate the heterogeneous nodes and resources in SAGIN. In this context, network slicing is a viable technique for efficient heterogeneous resource scheduling and management [295], [296].

B. Feasibility

1) Deployment: There are relatively few studies on the deployment of ENs, including edge servers or IoT devices in resource scheduling. The geographical location of ENs has a great impact on resource scheduling. Enlarging the service range of ENs can effectively improve edge resource utilization and effectively improve resource scheduling utility [30]. In many cases, the users are mobile, and ENs’ deployment will be more complex. Therefore, future research can consider the deployment of ENs when designing resource scheduling mechanisms.

2) Management: For the edge, scheduling computation tasks of users at the infrastructure is mostly limited to theoretical research. The technical issues on the implementation have not been well explored. Besides, the scalability of resource scheduling algorithms should be taken seriously. With the rapid expansion of users’ scale, the resource scheduling scheme is required to achieve flexible deployment and rapid configuration [32]. Serverless computing has become a popular architectural alternative for building and running up-to-date applications and services [152]. Serverless applications allow developers to focus on the code rather than on infrastructure configuration and management, which can speed up service provisioning and provide more efficient scaling [297]. The serverless computing architecture realizes the automatic scalability of services, pay-by-value, and automated high-availability management, which provides a powerful and convenience orchestration framework to schedule and manage edge resources. However, research on applying serverless architectures to edge computing is in its infancy, and many problems remain unsolved. Therefore, more attention need to be paid to resource scheduling research based on the serverless edge architecture.

C. Security and Privacy

1) System-Level: In the existing resource scheduling research, security and privacy issues have not been appreciated and fully explored. In resource scheduling, the multi-layer architecture of edge computing makes the edge system vulnerable to hostile attacks [182]. A system failure of an edge node or a failure caused by attacks may threaten the reliability and robustness of the entire edge system, thus making the resource scheduling meaningless. Therefore, efforts are required to put into the fault tolerance research of edge systems in resource scheduling. Specifically, system robustness enhancement mechanism and intrusion detection strategy need to be developed.

2) Service-Level: In the existing research on computation offloading and service provisioning, the following issues are generally not considered: whether the offloaded edge node can be trusted, how to ensure that users can authorize the edge services, and how to protect the privacy of the data generated by the edge service. Therefore, designing authentication mechanisms for the users covered by a specific edge node is needed. Besides, the privacy module is also required for the edge data center to improve the trustworthiness of edge services.

3) Data-Level: In the process of resource scheduling, especially computation offloading, data collected by the edge or shared with IoT devices involve much private information. In the existing research, the user data, the interaction data between ENs, and the computing data at the edge are unconditionally trusted and easily accessible [29]. However, in real application scenarios such as smart home and smart health, these data involves privacy and even commercial secrets of users, and can be easily leaked during transmission and processing, causing huge losses [31], [298]. Therefore, more works are needed to focus on designing trust mechanisms and privacy preservation policies for the edge and users.

D. Dynamics

In resource scheduling, users’ mobility is a thorny challenge. In various application scenarios, users’ mobile characteristics have not been well explored in current research,
and most studies just conduct idealization and ignore this characteristic. The frequent mobility of users has a significant impact on task offloading and cache provisioning. The offloading decision and cache decision at the current moment may not be applicable to users at the next moment, or even users have moved out of the service range of the edge node [10]. Therefore, incorporating the trajectory prediction of users into resource scheduling studies can effectively improve the users’ QoS. Moreover, designing the mobility management policies to enable users to access ENs seamlessly can improve the service stability.

E. Joint Scheduling of Communication, Computing, Storage (CCS) Resources

Task data should be received by processing nodes and cached in the data queue, waiting for processing to accomplish the offloaded tasks. The caching and queuing process is complicated and also very important for real-time task processing. However, in most existing works, the total task processing time is considered as the sum of local processing time, transmission time, and offloading processing time, ignoring the caching and queuing process. Besides, most studies on scheduling cache resources focus more on caching popular content at the network edge to improve hit ratio and avoid duplicate transmissions of the same content, thus improving users’ QoE [25], [299]. A few works have been done to consider combining the joint allocation of communication and computing resources. Therefore, future work on joint scheduling of CCS resources should take the research further forward by considering the caching and queuing process.

F. Evaluation

1) Workload: The workload of users’ requests has a non-negligible impact on resource scheduling. The requests from users are generally assumed to obey a specific distribution (e.g., Poisson distribution) in the current evaluation. Furthermore, the scheduled task’s CPU, memory, and storage requirements are treated theoretically and idealistically without considering real system performance. However, in the real environment, the peak situation of workload may put abnormal pressure on edge resources and even cause users’ tasks to fail [29]. Therefore, resource provisioning based on workload prediction is an urgent problem for SPs. Also, for reliable service, a good load balancing strategy needs to be designed.

2) Test Environment: The performance evaluation of scheduling algorithms in current research is generally performed using simulation tools, including professional simulators for edge computing such as iFogSim [300], EdgeCloudSim [301], and MyiFogSim [302], and general simulation platforms like MATLAB. Few studies evaluate their algorithms in real edge systems. Effort is required to focus on the feasibility of scheduling algorithms in real systems, e.g., designing testbeds or prototypes for evaluation.

VII. CONCLUSION

In this survey, we conduct a systematic and comprehensive review of resource scheduling in edge computing. First, we lay the groundwork for the entire overview by elaborating on two fundamental questions of why resource scheduling is needed and what exactly resource scheduling refers to in edge computing. Second, we present the architecture and different collaborative manners for resource scheduling. Third, an in-depth overview of research issues and research techniques in resource scheduling is presented, which is the prominent effort of this survey. Regarding the key research issues, we first introduce a unified offloading model for edge computing. Then we summarize the current works from three research aspects including computation offloading, resource allocation, and resource provisioning. Regarding the key techniques, based on two operation modes, namely, centralized and distributed modes, the state-of-art works are investigated and explicitly categorized. Also, we summarize six performance indicators that frequently appear in the surveyed literature. Fourth, some typical application scenarios involved in resource scheduling are introduced. Finally, for resource scheduling in edge computing to be investigated extensively and deeply, we shed light on the current research bottlenecks and challenges and look forward to more research investment in promising research directions.

REFERENCES

Authorized licensed use limited to: SOUTHWEST JIAOTONG UNIVERSITY. Downloaded on November 22,2021 at 02:40:41 UTC from IEEE Xplore. Restrictions apply.


Authorized licensed use limited to: SOUTHWEST JIAOTONG UNIVERSITY. Downloaded on November 22,2021 at 02:40:41 UTC from IEEE Xplore. Restrictions apply.


X. Cao, J. Xu, and R. Zhang, "Mobile edge computing for cellular-

Z. Zhou, J. Feng, Z. Chang, and X. Shen, "Energy-efficient edge computing service provisioning for vehicular networks: A consensus

A.-L. Jin, W. Song, P. Wang, D. Niyato, and R. Yu, "Social welfare maximization in container-


Z. Zhou, J. Feng, Z. Chang, and X. Shen, "Energy-efficient edge computing service provisioning for vehicular networks: A consensus


Q. Luo, C. Li, T. H. Luan, W. Shi, and W. Weigang, "Self-learning based computation offloading for Internet of Vehicles: Model and algo-


Quyuan Luo received the Ph.D. degree in communication and information system from Xidian University, Xi’an, China, in 2017, where he currently serves as the Lead Guest Editor for the IEEE/ACM Connected Health: Applications, Systems and Engineering. In 2019, he served as the Lead Guest Editor for the Edge Computing special issue on the prestigious PROCEEDINGS OF THE IEEE. He is an ACM Distinguished Scientist.

Shihong Hu received the bachelor’s degree in communication engineering from Jiangnan University in 2016, where he is currently pursuing the Ph.D. degree with the School of Artificial Intelligence and Computer. She has been a Visiting Scholar in Prof. W. Shi’s MIST Lab for research on resource scheduling in edge computing project, Wayne State University, USA, from 2019 to 2020. Her research interests include wireless sensor networks and edge computing.

Changle Li (Senior Member, IEEE) received the Ph.D. degree in communication and information system from Xidian University, Xi’an, China, in 2005. He conducted his Postdoctoral Research in Canada and with the National Institute of Information and Communications Technology, Japan. He has been a Visiting Scholar with the University of Technology Sydney and is currently a Professor with the State Key Laboratory of Integrated Services Networks, Xidian University. His research interests include intelligent transporta- tion systems, vehicular ad hoc networks, and wireless sensor networks.

Guanghui Li received the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2005. He is currently a Professor with the Department of Computer Science, Jiangnan University, Wuxi, China. He has published over 70 papers in journal or conferences. His research was supported by the National Foundation of China, Zhejiang, Jiangsu Provincial Science and Technology Foundation, and other governmental and industrial agencies. His research interests include wireless sensor networks, fault tolerant computing, and nondestructive testing and evaluation.

Weisong Shi (Fellow, IEEE) received the B.S. degree in computer engineering from Xidian University, Xi’an, China, in 1995, and the Ph.D. degree in computer engineering from the Chinese Academy of Sciences, in 2000. He is a Charles H. Gershenson Distinguished Faculty Fellow and a Professor of Computer Science with Wayne State University, USA, where he directs the Mobile and Internet Systems Laboratory and Connected and Autonomous Driving Laboratory, investigating performance, reliability, power- and energy-efficiency, trust and privacy issues of networked computer systems, and applications. He is one of the world leaders in the edge computing research community and published the first book on Edge Computing. His paper entitled Edge Computing: Vision and Challenges has been cited more than 1700 times. In 2018, he led the development of IEEE Course on Edge Computing. He is the Founding Steering Committee Chair of the ACM/IEEE Symposium on Edge Computing and the IEEE/ACM Connected Health: Applications, Systems and Engineering, in 2019, he served as the Lead Guest Editor for the Edge Computing special issue on the prestigious PROCEEDINGS OF THE IEEE. He is an ACM Distinguished Scientist.