A Survey of Computation Offloading With Task Types

Siqi Zhang, *Graduate Student Member, IEEE*, Na Yi, *Member, IEEE*, and Yi Ma, *Senior Member, IEEE*

Abstract— **Computation task offloading plays a crucial role in facilitating computation-intensive applications and edge intelligence, particularly in response to the explosive growth of massive data generation. Various enabling techniques, wireless technologies and mechanisms have already been proposed for task offloading, primarily aimed at improving the quality of services (QoS) for users. While there exists an extensive body of literature on this topic, exploring computation offloading from the standpoint of task types has been relatively underrepresented. This motivates our survey, which seeks to classify the state-of-the-art (SoTA) from the task type point-of-view. To achieve this, a thorough literature review is conducted to reveal the SoTA from various aspects, including architecture, objective, offloading strategy, and task types, with the consideration of task generation. It has been observed that task types are associated with data and have an impact on the offloading process, including elements like resource allocation and task assignment. Building upon this insight, computation offloading is categorized into two groups based on task types: static task-based offloading and dynamic task-based offloading. Finally, a prospective view of the challenges and opportunities in the field of future computation offloading is presented.**

Index Terms— **Computation task offloading, task type, energy, static task, dynamic task, offloading strategy.**

I. INTRODUCTION

NOWADAYS, with the rapid growth of computation-intensive and latency-sensitive mobile applications has significantly increased the demands on user equipment (UE) in terms of computation capacity and battery life (see [1], [2], [3]). Despite considerable enhancements for UEs, they still fall short of meeting these demands. Local execution of these applications on UEs could result in a range of issues, including processing errors for computational tasks and task timeouts [4], [5], [6], [7].

Computation offloading refers to the practice of relocating computational tasks from a device with limited resources to a more capable remote processing node (RPN), like a cluster, grid, cloud, or edge server [2]. Offloading the computation-intensive and latency-sensitive tasks from UE to RPNs with extra computation resource has been recognized as one of the feasible solutions to handle such issues mentioned above. The powerful computation resource of RPN can largely compensate the insufficient computation capability of the UE. Therefore, more researchers are focusing on computation offloading. As depicted in Fig. 1, the recent years have witnessed a dramatic increase in the number of publications on computation offloading. These publications cover a broad spectrum of scenarios, ranging from general systems to specialized areas such as smart transportation systems, robotics swarms, smart home systems [2]. This trend demonstrates the versatile application and growing interest in computation offloading across various fields.

Many research efforts have been paid to investigate computation offloading in various applications, i.e., autonomous vehicles [8], [9], [10], [11], [12], [13], [14], unmanned aerial vehicle (UAV) [15], [16], [17], [18], [19], cloud gaming [20], [21], [22], and robot swarm [23], [24], [25], [26], [27], [28], [29]. Examples of computation offloading applications are also shown in Fig. 2. The left panel of Fig. 2 illustrates an example of an autonomous vehicle. In this scenario, the vehicle transmits sensor data to a nearby base station, which is linked to an edge server. This server analyzes the data and sends back the processed results to the vehicle through the base station, facilitating autonomous driving. The right panel of Fig. 2 depicts an example of robot based task offloading. Here, the robot is responsible for sending sensor data to the computer. After processing this data, the computer generates control signals and sends them back to the robot.

Various challenges exist in the computation offloading across different scenarios. One such challenge arises from the dynamic task arrival rate, often experienced in the autonomous vehicle scenarios. This issue can hinder the efficient and effective allocation of resources during computation offloading. If not managed properly, it can lead to resource underutilization or overloading, which in turn can result in increased costs and potential system failures [8]. Another significant challenge in computation offloading research is the network congestion and considerable resource waste caused by transmitting large volumes of redundant data. This issue is particularly prominent in scenarios such as cloud gaming, robot swarms, and collaborative UAV applications [23], [29], [30].

The architecture of computation offloading also received much attentions. Mobile cloud computing (MCC) was proposed to be used to help the computing task offloading in [31]. However, cloud servers are generally far away from the UE.
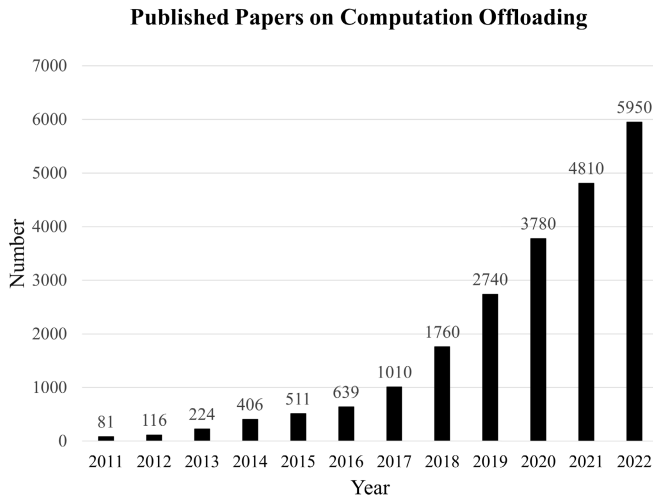
Published Papers on Computation Offloading

Fig. 1. The number of computation offloading related publications per year from 2011 to 2022 (data collected from Google Scholar with key words "computation offloading").

When UE needs to transmit the data to the cloud for processing the task, although the computation resource at the cloud is sufficient, the latency caused by transmission is considerable. Moreover, due to the characteristic of centralized processing of MCC, a large amount of data is offloaded from the UE to the cloud. This will cause excessive network overload. To solve this problem, another solution was proposed to move the RPN to the edge of the network.

One of the early schemes by bringing RPN closer to the UE was to use mobile edge computing/ multi-access edge computing (MEC) [32]. European Telecommunications Standards Institute (ETSI) released the MEC standardization in 2014 [33]. The purpose of MEC is to move the computation to the edge of network and bring the RPN closer to the UE. It can achieve lower communication latency than MCC. Another scheme, fog computing (FC), was introduced in 2012 by Cisco to enable the processing of the applications on billions of connected devices at the edge of the network [34]. It uses edge devices which are closer to UE to process a large number of tasks, so that it is more suitable for the context of Internet of Things (IoT) by comparing with MEC. Moreover, such as cloud-fog, cloud-MEC, these multi-level computation offloading architectures were also often considered [35]. These architectures can maximize the use of the advantages of different computing methods. Multi-level computation offloading was used when the number of tasks is large and the requirements (latency, computation capacity) of the tasks vary greatly [36].

Another focus of computation offloading research is the optimization, which mainly focused on task allocation and resource allocation [37]. To explore the diversity brought by fading channels [38], [39], [40] and multiple RPNs, two folds of work has been proposed to investigate task allocation and resource allocation correspondingly in [41] and [42]. In multi-MEC scenario, authors considered allocating different tasks to different MECs to achieve the best energy efficiency [41], [43]. In [44] and [45], authors considered splitting the task into subtasks and then allocating the subtask to different RPNs to gain energy efficiency.

For resource allocation, resources can be divided into two main categories: communication resource and computation resource [46], [47], [48]. Bandwidth, communication time, signal frequency, relay nodes, and transmission power are considered as the main communication resource which need to be optimized in computation offloading to improve the performance [49], [50], [51], [52], [53]. For computation resource, it includes the computation resources of UE and the computing resources of RPN. Computation resource allocation is common in multi-RPN scenarios and in partial offloading scenarios [54], [55]. In addition to these two major aspects, there are other aspects for various optimization methods and optimization objectives. For example, the splitting ratio needs to be taken into account when task splitting is used [45]. Time allocation for energy-harvesting process needs to be studied while energy-harvesting is considered [56]. Moreover, data correlation needs to be considered while optimizing the transmitted data [57].

It is also worth pointing out that the data generation of tasks has a significant impact on task offloading [58]. According to the features of data generation, the tasks can be divided into two types in our viewpoint, i.e., static task and dynamic task (for detailed definition, please see Section II-A.1). Dynamic task offloading often suffer higher complexity in optimization than static task offloading (e.g., dynamic changes in data size [57], dynamic arrival of tasks [59], dynamic computation density [60]). Moreover, in some cases, static tasks can be converted into dynamic tasks through optimization. For example, in [45], authors converted the task from static task into dynamic task by considering the task splitting based on the channel state information (CSI) and UE computation capacity. In [57], authors considered the redundancy removal to dynamically remove part of the data to change the task from static task into dynamic task. Although the task type changing will increase the complexity of the computation offloading algorithm, it can improve the resource efficiency, i.e. energy efficiency.

To the best of our knowledge, there is no comprehensive survey has been provided according to the different types of task for computation offloading. Therefore, in this paper, we propose a novel classification method, and employ the historical method to survey the corresponding work over these two categories. The rest of this paper is organized as following: an overview of computation offloading is given in Section II, which includes the analysis of the computation offloading process. Next, the related survey papers about computation offloading are introduced in Section III. Then, a detailed survey is conducted for computation offloading based on static task in Section IV and dynamic task from energy perspective in Section V respectively. Finally, in Section VI, we present a discussion, forecasting the possible evolution and the corresponding challenges that computation offloading will have in the years to come, and the conclusion.

II. OVERVIEW OF COMPUTATION OFFLOADING

The overview of computation offloading is given in this section. We review the computation offloading process including the current state-of-the-art (SoTA) and challenges in the

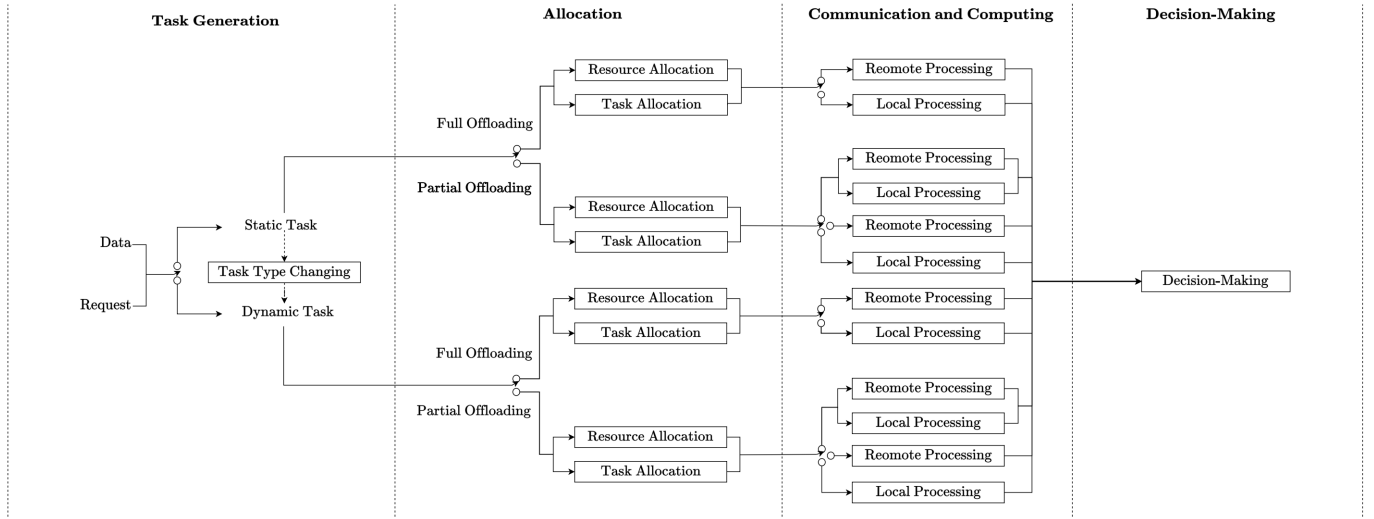Fig. 2. Examples of computation offloading applications (Provided by 5GIC & 6GIC, university of surrey).

Fig. 3. Computation offloading process.

different steps. Then, we summarize the current hot directions in computation offloading related research.

A. Computation Offloading Process

In our work, the computation offloading process is divided into four steps: task generation, allocation, communication and computing, and decision making as shown in Fig. 3.

1) Task Generation: The step one of computation offloading is task generation. In the computer science domain, a task is originally defined as the basic programming unit controlled by the operating system [115]. This definition is not suitable for the computation offloading system, since the relationship between the various factors formed the task is not described. Thus, based on [115], the definition of task is redefined as: a task is the basic programming unit controlled by the operating system, and it is formed by the request, data. Once a UE receives the request, it has to start using the corresponding functions[1] to process the corresponding data (or offload the corresponding data). The generation of data can be divided into periodic and non-periodic generation, and generation of request can also be classified in this way. Thus, the task is only

generated periodically if both data and request are generated periodically.

According to the relationship among tasks, data, and requests, tasks can be categorized as either dynamic or static. If the task of UE is static task, the following two conditions need to be met at the same time: tasks are generated periodically; the data size and CPU cycles requirement of the corresponding task in different time will not change. If the two conditions of the static task are not met simultaneously, the task of UE is defined as dynamic task. Furthermore, based on the proposed concept of task type, dynamic tasks typically arise in two manners. The first is when the tasks are directly generated during the task generation process (referred to as the first type of dynamic task). The second is when dynamic tasks are derived by intervening the generated static tasks (referred to as the second type of dynamic task). Common intervening methods include task splitting [45], [72] and redundancy removing [57]. The task obtained by splitting a static task can be seen as a dynamic task, because the ratio of the splitting in different time slots may be different, dynamically changing the data size of each subtask. Redundancy removing can also be assume as an effective intervention of changing static tasks into dynamic ones, this is because the data size for each task will vary according to the correlation degree. Task generation

---

[1]When function does not appear with objective (objective function), it refers to the function used to process data.

TABLE I

PAPERS CORRESPONDING TO EACH CATEGORY

		Static Task Generation	**Dynamic Task Generation**
Optimization Objective	Energy Saving Maximization		
	1) Energy consumption of UE only	[61]-[66]	[29], [56], [57], [59], [60], [67]-[74]
	2) Energy consumption of multiple entities (including various combinations of UE, relay node, edge or cloud server)	[8], [19], [75]–[78]	[79], [80]
	Latency Minimization	[10], [22], [81]-[85]	[86]–[89]
	Multi-objective (energy, latency, and reliability)	[15], [17], [23], [44], [90]–[101]	[25], [45]
Collaboration Strategy	Non-Collaboration	[22], [61]–[65], [75], [76], [81], [83]–[85], [90]–[94], [96]–[103]	[16], [30], [56], [57], [59], [60], [67]-[74], [86]-[89], [104]–[107]
	Collaboration	[8], [15], [17], [19], [23], [43], [66], [77], [82], [95], [108]–[110]	[111]–[114]
Offloading Strategy	Full Offloading	[8], [10], [15], [43], [63], [65], [75], [81], [82], [84], [91], [93], [96]–[102], [109]	[11], [16], [45], [56], [59], [60], [68], [71], [86], [104], [105]
	Partial Offloading	[44], [61], [62], [76], [77], [83], [85], [90], [92], [94], [103]	[25], [67], [70], [72]–[74], [87], [88], [106], [107], [111]
Application Domain	IoT	[62], [63], [88], [95], [96]	[73]
	Autonomous vehicle	[8], [19], [63], [66]	[11], [69]
	Robotics	[23]	[25], [29]
	AR and VR	[20], [22]	[21], [106]

plays a pivotal role within transportation systems [9], [116], [117], [118], [119]. In such systems, tasks might involve route planning for vehicles, traffic flow prediction, or accident detection. For instance, when an autonomous vehicle needs to navigate from one point to another [113], it generates a task to plan the optimal route. This task can take on a static form, with a predetermined route set before departure, or it can be dynamic, adapting based on real-time traffic updates and changes in the environment. Furthermore, consider smart traffic signal systems in urban settings. These systems may generate tasks aimed at forecasting future traffic flow, allowing for more efficient adjustments in the timing of traffic lights.

In SoTA studies exploring computation offloading with static tasks [8], [10], [120], [121] (for further references, see Table I), certain parameters such as task generation frequency, data size, and their associated CPU cycles remain constant. This fixed state simplifies the optimization process by reducing its complexity. Conversely, in SoTA studies on computation offloading with dynamic tasks [57], [122], [123], [124] (for further references, see Table I), due to the dynamic nature of these tasks, further optimizations are required to design the optimization function (e.g., optimization in queuing problems [122], task splitting ratios [125], and data removal ratios [57]). Optimizing computation offloading with dynamic tasks offers greater flexibility but also introduces increased complexity. Furthermore, converting a static task into a dynamic one through intervention significantly enhances its flexibility in resource and task allocation. However, this transformation also substantially amplifies its complexity.

Most of the computation offloading related research is based on the properties of the generated task (i.e., static task or dynamic task) to directly perform resource allocation, task allocation, and other optimizations [2]. Therefore, most of their optimization starts from the step two of computation offloading. When changing the task type is taken into account, it can be assumed that the corresponding optimization starts from the first step of computation offloading.

2) Allocation: The second step is allocation, and it includes the resource allocation and task allocation. This step is the focus of current computation offloading related research [2]. We list some key points in Table I, and introduce them in detailed in the following part.

Optimization objective is one of the important key points needed to carry out the allocation. Different scenarios and applications have various restrictions and requirements for tasks, so their corresponding optimization objectives will not be the same. For example, ultra-reliable low latency edge computation need to consider the reliability; energy consumption of cloud server in MCC will not be considered usually. Based on [1], [2], and [35], the main optimization objectives are summarized as follows.

The first optimization objective is energy saving maximization. The energy consumption is the focus of current research

on computation offloading, and it includes energy consumption of the computing and communication [2]. The energy consumption caused by computing mainly includes energy consumption of local computing and RPN computing. The energy consumption caused by communication is primarily composed of communication (downlink and uplink) between UE and relaying node [95], different UEs [66], UE and RPN [126]. Moreover, when a UE is on standby or a task is in the queue, the UE also need to consume energy. It is called the energy consumption caused by waiting, and because this part of the energy consumption is tiny (comparing with energy consumption of communication and computing), most researchers directly ignore it [35].

The optimization objects will be different for different scenarios and applications. Typically, in scenarios where tasks are offloaded to MCC or MEC servers with a continuous energy supply, researchers do not consider the energy consumption of these servers, as noted in several studies [18], [59], [60], [61], [62], [63], [67], [68], [69], [70], [71], [72], [75], [81], [90], [91], [92], [93], [94], [95], [127], [128]. However, there are many research works that consider the energy of edge and cloud servers in the optimization [78], [79], [80], [129], this is because cloud service providers and data centers need to optimize their energy usage to reduce their environmental impact and lower operating costs. Moreover, in some cases where fog computing is considered based on IoT network, because some computing nodes themselves do not have continuous energy supply, it is necessary to consider the energy consumption of fog nodes [76], [109], [130]. In some scenarios where relaying node is used to help UE and RPN to communicate, the energy consumption of relaying node will also be taken into account, such as using the UAV worked as the relaying node [18], [131].

In some scenarios, the data transmitted by the UE to the RPN is much larger than the result of the task received by the UE from the RPN [56], [75], [96], [132]. Therefore, the energy consumption of downlink transmission is often disregarded in optimization processes [29], [35], [56], [75], [96], [132], [133]. However, there are some applications, such as AR, because UEs need to download a large amount of data from the RPN (RPN with content provider), the energy consumption of downlink is the main optimization object, so the corresponding energy consumption for such applications cannot be ignored [22], [134].

The second optimization objective is latency minimization [20], [21], [22], [82], [83], [86], [87], [88], [88], [135], [136], [137], [138], [139], [140], [141], [142]. Although the viewpoint of our survey is energy consumption, the studies with optimization objective of latency minimization would also use energy consumption as a constraint for optimization, so it is also necessary to investigate it. The latency mainly includes computation latency and communication latency [143], [144]. The computation latency also includes the latency of local (UE) computing and remote (RPN) computing. Communication latency includes the latency of communication (downlink and uplink) between different UEs, UE and relaying node, UE and RPN, RPN and RPN. When the data size of the

feedback signal from RPN to UE is small, most studies do not take into account the time cost of downlink [54], [83], [84], [88], [145], [146], [147].

In addition to the two mainstream optimization objectives of energy saving maximization and latency minimization, there are many other optimization objectives that are being explored in computation offloading research. These optimization problems often involve multiple objectives and trade-offs between factors, with energy consumption being a critical factor that is often considered. The most researched optimization problem for multiple objectives is the trade-off between latency and energy consumption [18], [44], [64], [85], [85], [90], [91], [92], [93], [94], [95], [97], [98], [99], [100], [105], [125], [128], [148], [149]. In their work, they weighted energy consumption and latency, and designed different algorithms to minimize the weighted sum. As an extension to the issue of time and energy costs, other costs are considered, such as the cost of renting an RPN [15], [101].

Furthermore, reliability is often an important factor to be considered in computation offloading. The considered reliability includes communication reliability and computing reliability [106]. An unreliable computation offloading system is meaningless, and in some studies, researchers assume that the errors caused by communication and computing do not affect the results of the task [57], so they can ignore the issues caused by reliability. However, for computation offloading that works in the ultra-reliable low latency scenario, the factor of reliability cannot be ignored, so most research regards reliability as a constraint to limit optimization function in this scenario. Moreover, with increasing demands on reliability, more and more researchers are now considering the trade-off between reliability and energy consumption in computation offloading [45], [106].

Collaboration strategy is another key point to carry out the allocation. Different collaboration strategies have different optimization ideas. There are two types of collaboration in computation offloading, they are communication collaboration (the collaboration for communication between different devices) [150] and computing collaboration (the collaboration for computing between different devices) [35], [43]. Thus, the collaboration strategy for computation offloading can be divided into three different categories, they are non-collaboration, computing collaboration and communication collaboration. Different collaboration strategies have their advantages and disadvantages based on their features.

The first collaboration strategy is non-collaboration. When there is no collaboration between different UEs and no collaboration between different PRNs, the corresponding collaboration strategy is called non-collaboration [22], [61], [62], [63], [75], [76], [81], [83], [90], [91], [92], [93], [94], [96].

The second collaboration strategy is computing collaboration. Computing collaboration is divided into two categories, they are RPN computing collaboration [43], [95], [109], [151] and UE computing collaboration [23]. RPN computing collaboration computation offloading refers to different RPNs collaboratively processing tasks offloaded by UE. When computation offloading works in a complex multi-user, multi-task

system, it is likely that the task requirements will vary significantly. Some tasks may be highly computation-intensive and require significant computing resources, while others may have strict latency constraints and require real-time processing. Tasks that are computationally intensive but do not demand on low latency are better suited to be offloaded to the cloud server for processing; tasks that do not require much computation but require extremely low latency are better suited to be offloaded to MEC servers or fog nodes for processing; tasks with high computation and low latency requirements are more suitable to be offloaded to MEC servers for processing. When the tasks with different requirements arrive at the same time, it is difficult to meet all requirements of different tasks if there is only one RPN. Therefore, this motivate researchers to consider this computing collaboration computation offloading [2]. The most common architecture of computing collaboration computation offloading is MEC-cloud [43], [95], [109], [151], [152]. In [151], Guo et al. used MEC-cloud collaboration for IoT over the fiber wireless networks to achieve the goal of energy consumption minimization. In [109], Zhao et al. focused on the collaboration computation offloading in the vehicular network, which is based on cloud-assisted mobile edge computing. The authors maximize system utility in their study by considering the task and resource allocation problems under latency constraints. UE computing collaboration computation offloading refers to different UEs collaboratively process tasks such as [110], [111], [113], and [153]. This collaboration often occurs in vehicle scenario. Allocation of UE's idle computing resources is the focus of research [154].

The third collaboration strategy is communication collaboration. Communication collaboration is also divided into two categories, they are communication collaboration between UEs [95] and communication collaboration between RPNs [66]. The communication collaboration between UEs refers to the collaboration between different UEs to transmit the data collaboratively, and it also be called multi-hop collaboration [150]. Many studies on computation offloading assume that all UEs can connect directly to the RPN through the wireless network, which is referred to as single-hop computation offloading [95], [155]. However, in practice, UE may experience network connectivity issues, even out of the coverage of RPN, and that may not be able to directly connect to the RPN. Thus, in some cases, some UEs also need to act as relay nodes to help other UEs to transmit their data to the RPN, and it is called communication collaboration. For example, when the deployment of UEs is very dispersed, some UEs will be very close to the PRN and some UEs will be very far away from the PRN (even beyond the communication range). In this case, if single-hop communication is used, there is a high probability that the computation offloading of those UEs that are very far from the RPN will fail, leading to the failure of the computation offloading system. Therefore, this motivate researchers to consider the communication collaboration between UEs, those UEs close to the PRN need to act as relay nodes to help those UEs far from the PRN to offload their tasks in addition to offloading their own tasks [18], [23], [23], [66], [112], [156], [157], [158], [159]. These studies involve a lot of

relay node (UE) selection problems. Communication collaboration between UEs is often used in the scenarios involving UAVs, robots, vehicles [15], [17], [19], [23], [24], [131], [160]. Funai et al. [157] studied trade-off problem in terms of computation delay and network lifetime in a cooperative multi-hop ad hoc network. Müller et al. [161] investigated the problem of minimizing energy consumption for computation offloading in multi-hop wireless network. Hong et al. [23] studied the communication collaboration between different robots in robot swarms. The communication collaboration between RPNs refers to the collaboration between different PRNs to transmit the data collaboratively [66]. In the context of communication collaboration between RPNs, researchers focus on the problem of RPN selection for data transmission, i.e., which RPNs are used for the communication (as relay node) so that the processed data can be received by the moving UE [66].

Offloading strategy is the third point to carry out the allocation. The offloading strategy can be divided into two types, full offloading and partial offloading.

In full offloading, there will be two situations. The first situation of full offloading is that all tasks will be offloaded to the RPN for processing. It means that all the tasks need to be offloaded whatever the environment (wireless channel conditions, computation resource availability) changes, such as [45], [59], and [82]. This kind of situation is often considered in the sensor network because of the limited computation capacity of the local device. The second situation of full offloading is that UE offloads all the tasks to the RPN, or process all the task locally [75], [81], [108]. It means that all the tasks (generated in the same time slot), which belong the same UE, will be processed in the same terminal together. Furthermore, when the UE only has one task and does not consider task splitting [75], then there are only two possible outcomes: either offload the task or not offload it. Thus, this case is identified as second situation of full offloading based on the definition that mentioned earlier.

In partial offloading [25], [61], [62], [67], [70], [72], [76], [87], [88], [90], [92], [94], UE can dynamically determine which tasks (for multi-task cases) or which part of task (if task splitting is considered) are offloaded to the RPN based on the channel state information, computation capability of UE, the computation capability of the RPN. Unlike the second case in full offload, tasks of the same UE in the same time slot can be processed separately in different places. Moreover, considering the partial offloading with task splitting from application model. The applications can be divided into two types. They are data partitioned oriented application and code partitioned oriented application [2]. For data partitioned oriented application, the corresponding tasks can all be divided into two parts of any size [58], [162]. On contrast, for code partitioned oriented application, the corresponding task cannot be split into any size [163], and it requires the selection of the code to be offloaded (the data for the same code cannot be separated) [164]. Thus, data partitioning oriented application is more flexible than code partitioning oriented application. Moreover, considering the partial offloading with knowledge

on the amount of data to be processed. The applications can be divided into two types. They are data amount known based application and continuous-execution application [2]. For the data amount known based application (such as the studies in [163]), the amount of data which is need by the application is known before offloading process. For the continuous-execution application (such as the studies in [165], [166]), it represents the application that the amount of data which is need by the application is unknown before offloading and it is hard to estimate the data amount requirement. Thus, it will largely increase the difficulty of partial offloading comparing with the data amount known based application.

With the same optimization objective and collaboration strategy. Full offloading has a lower complexity in step two of computation offloading process [2], since in the first situation of full offloading, it is only necessary to consider offload all the task to RPN; in the second situation of full offloading, it is only necessary to consider the binary offloading further (i.e. to offload or not to offload). However, in partial offloading, researchers need to study which tasks of the UE need to be offloaded, which tasks need to be executed locally, whether tasks need to be offloaded to different RPNs. In general, the optimization algorithm for full offloading is simpler, but its performance is worse. The optimization algorithm for partial offloading is a bit more complex, but the corresponding performance is better [2].

3) Communication and Computing: The third step of computation offloading is communication and computing. The step three corresponds to the data transmission process and the data processing process. In some cases, there are other processes, such as energy-harvesting (EH) process. This step is actually the implementation of the optimization designed in step two and it is just listed here to ensure the integrity of the computation offloading process. Thus, in the following, the survey for this step will not be carried out.

4) Decision Making: After computing, UE needs to collect the task output (collect the results from UE itself, or download the results, such as the control signal, from RPN) to make the corresponding execution decision. Thus, there comes the step four, decision making. This step may involve task fusion or data fusion (e.g., in some cases where task splitting is considered, further data fusion and task fusion are required to get the final complete results). This step is generally not taken into account by researchers in the optimization process. It is just listed this step here to ensure the integrity of the computation offloading process, and the survey for this step will not be carried out in the following part.

From our survey results, most researches on computation offloading focus on the second step, i.e., allocation (resource allocation and task allocation). Recently, more and more studies focus on joint optimization of step one and step two (e.g., joint optimization of task splitting and resource allocation and task allocation, joint optimization of redundancy removing and resource allocation and task allocation). Therefore, in the following part, the work of these two steps will be surveyed in detail.

B. Application of Computation Offloading

Computation offloading has emerged as a pivotal technique in the realm of modern computing, especially in scenarios where devices have limited computational resources or energy constraints. Essentially, it involves transferring specific computational tasks from resource-constrained devices, such as smartphones or IoT devices, to more robust servers or cloud platforms. The primary objective is to enhance performance, conserve energy, and extend the battery life of the initiating devices. In the context of mobile cloud computing [91], smartphones and tablets, which often grapple with limited battery life and computational capabilities, can greatly benefit from offloading. Intensive tasks, such as video processing, augmented reality applications, and high-end gaming, can be seamlessly transferred to the cloud, ensuring smoother user experiences without draining the device's resources.

The IoT is another domain where computation offloading proves particularly beneficial [119]. For instance, in smart homes, devices like thermostats and security cameras can offload their data processing tasks to the cloud. This enables more efficient collaboration among various devices. Similarly, in the healthcare sector, wearable devices like heart rate monitors or glucose meters can offload their data for more sophisticated analysis in the cloud, enabling real-time feedback and potentially life-saving alerts.

Autonomous vehicles [117], equipped with a plethora of sensors, stand to gain significantly from offloading. By transferring data to edge servers or cloud platforms, these vehicles can benefit from real-time processing, which is crucial for safe navigation and decision-making on the roads.

In the field of robotics [29], robots operating in dynamic environments can offload complex computational tasks, such as image recognition or path planning, to more powerful servers. This offloading not only facilitates their adept response to environmental changes but also enhances coordination among different robots.

Augmented reality (AR) and virtual reality (VR) [22], which demand high computational power for real-time rendering, can offload these intensive rendering tasks to the cloud. This ensures that users enjoy a smooth and immersive experience without overburdening the device's native hardware.

C. Several Research Directions Related to Computation Offloading

In this subsection, several research directions related to the step one and the step two of computation offloading are introduce respectively. The current research directions on computation offloading with their corresponding papers are listed in Table II, and it is used to summarize the problems in different research directions.

1) Task Splitting: Task splitting is an important way to change the task type in the first step of computation offloading, and it can be jointly optimized with allocation (step two) to improve the performance of computation offloading.

Task splitting allows a task to be divided into multiple subtasks that can be allocated to different computation resources for parallel processing, which can reduce the overall execution

time and energy consumption. By jointly optimizing task splitting and allocation, the system can make use of both local and remote resources effectively to achieve better performance. For example, a computationally intensive task can be split into two subtasks, with one subtask executed locally on the UE and the other offloaded to an RPN, in order to balance the workload and reduce the overall latency and energy consumption.

Task splitting is an approach to improve the energy efficiency of computation offloading [44], [45], [61], [64], [88], [106], [136], [167]. It can change static tasks into dynamic tasks through splitting, so as to carry out more flexible resource allocation and task allocation. By using task splitting, tasks can be split into multiple subtasks,[2] which can be allocated to multiple RPNs for processing. It can improve the utilization of communication resources and computing resources to achieve energy-saving or latency reduction. There are two existing task splitting methods for computation offloading systems. The first method is to split the task at any ratio with the assumption that there is full granularity in data partition of the task. The second split method is to split the task by using code/source partitioning. This means that task splitting is no longer arbitrary (not in any ratio anymore) and it has a certain splitting ratio (based on the attribute of code/source). In [44], [45], and [88], the authors gave the assumption that there is full granularity in data partition. Thus, the considered task could be partitioned into subtasks of any size. However, this way of splitting is idealistic, so that in [61], [64], [106], [136], and [167], the authors split the task according to the topological structure of task, such as code, function.

Two different task splitting methods have been introduced. In addition, task splitting can also be classified according to the number of subtasks after splitting, and there are two different types, they are splitting one task into two subtasks and splitting one task into more than two subtasks. When considering splitting a task into two subtasks [72], it is common to consider that one subtask will be processed on the UE and one subtask will be processed on the RPN. In this consideration, a UE only need to offload one of the subtasks to RPN during an offloading process, making split-ratio optimization a critical aspect of the optimization problem. If the task is split into multiple subtasks [45], there is a more complex problem of multi-RPN node selection involved, in addition to determining the split ratio. Although this is more complexity, it provides greater flexibility and can lead to better performance.

According to the relationship between subtasks, task splitting can also be divided into two types, i.e., parallel splitting and serial splitting. The consideration in [57] is parallel splitting, and the subtasks after splitting can be processed serially or in parallel. In addition, authors [45] considered serial splitting, and subtasks must be processed in a fixed order according to the topology of the corresponding task. The consideration in [106] is mixed splitting, which combines parallel and serial splitting.

Different tasks/subtasks may be independent of each other or related to each other, as well as the subtask. When they are completely independent of each other, they can be processed in parallel, and there is no need to consider the sequential problem. When they are related to each other, it will involve the problem that the optimal offloading path selection is restricted by the topological structure of the task [61]. Such issues often appear in the allocation of subtasks.

2) Redundancy Removing for Computation Offloading: Redundancy removing is another way to change the task type in the first step of computation offloading. The data considered in computation offloading can be divided into two categories: correlated data and independent data. When considering correlated data, it means that the loss of certain data does not affect the processing of the corresponding task [29], [168]. Therefore, removing this redundant data can help save energy or reduce latency, without affecting the correctness of decision making.

There are some studies about redundancy removing in computation offloading from the perspective of input data. In Section II-C.1, it is mentioned that different tasks or subtasks may be related to each other, the input data of the same task in the time domain may have a high correlation. Processing a task is actually an observation of the environment corresponding to the task. Under normal circumstances, the frequency of this observation is much greater than the change of the environment, which means that most observations are meaningless. This implies that the corresponding processing of the task is meaningless. Moreover, a task may be composed of multiple subtasks, so it may involve multiple observations of an environment from different angles. This kind of multiple observations of the same environment is also a kind of redundancy that can be removed. Nour and Cherkaoui [168] designed an experiment for object detection service based on the standard image dataset, ImageNet [169]. The experiment proves that this kind of redundancy exists in the real world, which provides a theoretical basis for the optimization method based on redundancy removing.

How to remove redundancy to reduce the repeated data transmission and repeated computation (include RPN computation and local computation) processes has become a new research hotspot [76]. In [57], Zhang et al. suggested that in the time domain, the input data of the same task at different times is repetitive on a large scale. Moreover, [30] and [102] conducted the similar work. The task splitting mentioned in Section II-C.1 is a good way to be further explored to search and remove the overlap between different tasks [57]. The work for redundancy removing in computation offloading is still relatively limited, but the performance improvement from the simulation results of existing papers is quite significant. As the specific work based on [57] and [168], Bellal et al. [89] proposed an efficient computing reuse architecture for edge computing called CoxNet. It enables the edge server to reuse the previous results while scheduling dependent incoming computing. Through evaluation based on real data sets, CoxNet can reduce task execution time by up to 50%. Furthermore, Nour and Cherkaoui considered this reuse architecture for IoT application [30].

Task splitting and redundancy removing are two ways to optimize computation offloading at the first step. Task splitting is initially designed to provide greater flexibility in

---

[2]In the whole paper, subtasks are split by tasks.

resource allocation and task allocation, thereby improving system performance. Data has a great impact [89] on computation offloading, but task splitting does not make good use of it. Redundancy removing is an approach to optimize tasks by exploiting data correlation in the task generation process. It uses the huge impact of data, but it still receives little attention by researchers. In the following parts of this subsection, the optimizations based on the second step of computation offloading are investigated.

3) Ultra-Reliable Low-Latency Computation Offloading:
Ultra-reliable low-latency edge computing is the new service for applications that demand high reliability and low latency, such as automation vehicle, industrial automation, and remote surgery. The ultra reliable low latency computation offloading has also attracted widespread attention, as seen in [45], [73], [87], [111], [170], [171], [172], and [173]. The conventional computation offloading system is designed based on average-based indicators, which cannot meet the requirements of ultra reliability and low latency. As a result, changing the conventional average-based design architecture and taking ultra-reliable and low-latency into account for optimization presents a major challenge for computation offloading. To address this challenge, some researchers considered task-queuing problem with dynamic task (task arrival rate can be assumed as a random variable) [73], [87], [111], [170], [171], [172], [173]. Liu et al. [72] considered the queuing problem with the reliability constraints, and their optimization objective is to minimize the energy consumption. They modelled the latency and reliability constraints by using task queue lengths based on the extreme value theory. The above work considered the queue problem, which always has a close relationship with the different arrival rates. This indicates that the research direction of task-queuing for ultra reliable low latency computation offloading focuses on dynamic tasks.

The above research primarily focuses on ultra reliable and low latency for the computation process. There is also a kind of research direction focusing the communication process for ultra reliable low latency computation offloading, so there is no excessive requirement on the task type. For example, Liu and Zhang [45] considered the block error rate as the communication error, and a threshold for communication error was set up to constrain the communication error of the computation offloading system. Several research directions mentioned above (Section II-C.1, II-C.2 and II-C.3) are more aimed at dynamic tasks or the changing from static task to dynamic task, and they are more targeted. There are also some research directions that do not emphasize the task type too much as follows.

4) Artificial Intelligence (AI) Used for Computation Offloading: When considering task allocation and resource allocation in computation offloading, a lot of the issues that need to be considered are NP-hard problems [174], [175]. For conventional computation offloading, many researchers try to solve them using heuristics [45], game theory [127]. However, these approaches are less flexible and rely on a specific environment for optimization. As a result, when the environment of the computation offloading system changes, the approaches may not achieve the optimum performance. Since AI methods can learn the near optimum response strategies for different situations from existing data, such methods can more effectively solve complex offloading decision-making and highly dynamic problems.

AI is used in computation offloading systems to determine resource and task allocation. It can work with both dynamic and static tasks and is used in the second step of computation offloading, without changing the type of task. The current AI technology has wide applicability, and the introduction of AI into computation offloading systems is considered a solution to the aforementioned problems [11], [174], [176], [177], [178], [179].

Reinforcement learning (RL) is commonly used in computation offloading. RL is a method of learning in dynamic systems that adjusts decisions based on whether the decision is positive or negative in different situations. It sets up a reward and punishment mechanism. Through continuous testing, it rewards and punishes a series of actions and then modifies its strategy. After such continuous adjustments, the UE can learn which actions should be chosen in order to achieve the best return in certain situations, as in [177], [178], and [179]. In addition, there are Q-learning [9], deep reinforcement learning (DRL) [180], [181], [182], and other techniques.

Beyond traditional approaches and reinforcement learning, meta-heuristic AI algorithms present another promising direction in the field of computation offloading. Meta-heuristic methods [183], inherently designed to find approximate solutions for complex optimization problems, align seamlessly with the challenges of computation offloading, especially when dealing with NP-hard problems. Infusing AI techniques into these algorithms enables more adaptability and efficiency. For instance, AI-augmented genetic algorithms [116] can dynamically adapt crossover and mutation rates based on the observed performance of offloading decisions. Similarly, AI-powered simulated annealing can use machine learning models to ensure a better convergence. One significant advantage of combining AI with meta-heuristics is the ability to generalize across various offloading scenarios. Unlike traditional methods that might be tailored to specific environments, AI-enhanced meta-heuristics learn from a wide range of data, enabling them to perform effectively even when system conditions change. Furthermore, with the vast amounts of data generated in computation offloading scenarios, deep learning models can be trained to identify patterns and insights that might be non-intuitive or too complex for traditional algorithms. This facilitates more intelligent initialization of meta-heuristics or even hybrid approaches that blend the strengths of various algorithms, guided by the insights of deep learning.

5) EH Used in Computation Offloading: Although there are many computation offloading algorithms for improving energy efficiency, which has maximized energy efficiency to a large extent, the process of computation offloading still requires energy and the problem of energy supply remains unsolved. Computation offloading need to be stopped when the battery of UE is used up. This can be overcome by charging the battery or using a larger battery. However, using the larger battery on mobile devices means increasing hardware costs, which is undesirable. Moreover, it may not even be possible

TABLE II
RESEARCH ON COMPUTATION OFFLOADING WITH CORRESPONDING PAPERS

Research Directions on Computation Offloading	Corresponding Papers
Energy-Harvesting (EH) for Battery Lifetime	[56], [65], [104], [105], [184]
Task Splitting for Allocation Flexibility	[16], [44], [45], [57], [67], [72], [76], [88], [106], [125]
Redundancy Removing for Data Efficiency	[30], [57], [89], [168]
Caching for Workload Reduction	[11], [68], [71], [101], [114], [185]
Ultra Reliable Low Latency Computation Offloading	[45], [59], [69], [72], [73], [87], [106], [111], [173]
Mobility Problem for Vehicle	[8], [25], [66], [137], [186]–[188]
AI for Offloading-Decision Making	[9], [11], [25], [73], [96], [105], [130], [177], [178] [175]
Collaboration Problem for UAV and Robot Swarm	[15], [17], [19], [23], [24], [29], [131], [160]

to charge the battery of the UE in some application scenarios. EH is a promising technology to solve these problems. It can capture environmentally recyclable energy [189], including solar energy, wind energy. EH, as a way of energy supply, can be used for the both static task and dynamic task. Its resource allocation process is a bit more complicated, because in most of consideration, communication and EH cannot be done at the same time, so time resources need to be allocated. To use this EH technology, Mao et al. [104] established a EH model for computation offloading. In their model, the EH process is modeled as a continuous energy packet arrives, and this is a basic way of EH.

You et al. [56] presented a different EH method, microwave power transfer (MPT), in computation offloading system. The base station (BS) transfers power wirelessly to UE, and UE's power comes entirely from MPT. In their work, authors assumed that local computing and MPT can work at the same time. However, the data transmission process and MPT cannot work at the same time (work in half-duplex transmission). Because the energy conversion efficiency of this MPT is not high, the energy obtained by the UE from the MPT is also very low. Therefore, the authors also mentioned in their paper that this EH method is more suitable for low-complexity devices such as wearable computing devices and sensors.

6) Mobility Problem in Computation Offloading: Mobility is one of the important features that cannot be ignored in computation offloading [186], especially in the computation offloading for vehicle [188]. This is therefore a feature that needs to be taken into account, whether the task is static or dynamic. Because of the mobility of UE, intermittent connections between UE and BS have become an often occurred state which can cause computation offloading to fail. According to our survey results, the mobility research in the field of computation offloading mainly focuses on task migration, and it includes selecting the appropriate RPN for offloading, and selecting the appropriate path for task (input data/output data) migration [103], [186], [187], [188], [190].

The location of the UE usually determines which RPN the UE needs to offload its task to, because the task is usually offloaded to the nearest RPN to achieve low latency requirement. Due to the limited coverage of the BS, the following situation may arise. A UE transmits data to an RPN, after the transmission, if the UE is not within the coverage of the BS corresponding to this RPN, the UE will not receive the results

corresponding to the transmitted data. This situation often occurs in the vehicle-to-everything (V2X) environment [188], because the coverage of road side unit and BS is not large, but the moving speed of the vehicle is very fast. Zhang et al. [66] proposed a predictive combination-mode relegation scheme for MEC computation offloading in a cloud-enabled vehicle network. In order to solve the problem of UE leaving the coverage area of the previous corresponding MEC due to mobility, the authors provided two approaches based on the movement prediction. The movement of UE is unpredictable and irregular in most complex environments. Due to the development of machine learning, it is now possible to predict a UE's estimated stay time in a given area and its movement habits. This provides a foundation for the development of this kind of computation offloading based on movement prediction [188].

7) Caching Used in Computation Offloading: The increasing demand for massive multimedia services over the mobile cellular network makes significant challenges to network capacity and backhaul links. The emergence of mobile edge caching and delivery techniques are promising solutions to cope with those challenges [185].

In conventional centralized mobile network architectures, it is typical for remote internet content providers to supply UE with necessary input data for tasks, a scenario frequently encountered in VR and AR [68] In scenarios where numerous UEs require identical input data or the same UE repeatedly requests the same data, the mobile network is burdened with transmitting redundant data. This repetition can lead to substantial network congestion and an inefficient use of network resources. Caching popular content at the edge of the network (it is not the Internet content provider, and it is close to UE) can cancel the repeated transmissions of the same content, which will significantly reduce the workload of communication and reduce energy consumption and decrease latency. In this case, the input data used to form the task does not come from the UE but the remote content provider. Another situation is that the UE itself generates data and the corresponding task, when the result of a task is cached in MEC server, the UE can download the result directly from the edge server [71], [185]. Thereby reducing the energy consumption and latency caused by communication (offload the task to MEC server) and computing. The advantage is that the task can be cached on the RPN in advance. After the task is generated,

as long as the UE finds that the task has been cached, it can directly download the corresponding results from the RPN. According to the four possible relationships across different task generation process mentioned in Section II-A.1, it is easy can find that for the same task at different times, if their corresponding data is different, their corresponding result may also be different. There is no point in caching such a task.

In this section, we have examined various research works related to computation offloading. A common observation is that the majority of these studies overlook the optimization of task types and often neglect the significance of data in this context. While some research on task splitting in computation offloading does consider task type optimization, their primary focus is generally on enhancing the flexibility of resource and task allocation. However, these studies tend to overlook the core of tasks, which is data. There are only a few studies that recognize the crucial role of data in computation offloading. This area of research is still nascent, suggesting that further exploration into data-centric computation offloading is both necessary and promising.

III. RELATED SURVEYS ON COMPUTATION OFFLOADING

Many studies related to computation offloading have been published. Thus, researchers have surveyed these publications from various perspectives. In this paper, we mainly focus on task type and energy consumption to survey the computation offloading related research. In addition to our proposed survey perspective, some researchers surveyed computation offloading related work from other viewpoints and perspectives.

Heidari et al. [191] surveyed computation offloading based on the IoT scenario and proposed a new taxonomy for computation offloading based on offloading decision mechanisms and overall architectures. The authors also pointed out the future research direction with the corresponding challenges of computation offloading used in IoT. Moreover, De Souza et al. [192] reviewed papers about computation offloading in vehicular environments.

Different types of RPN have their own advantages and disadvantages. Khan et al. [193] surveyed the work which considered the usage of MCC in computation offloading, and they pointed out the advantage and disadvantages of MCC. Mach et al. [2] did the literature survey for MCC in computation offloading from the perspectives of offloading decision, resource allocation, and mobility management. Deshmukh and Shah [194], Chalaemwongwan et al. and Kurutach, and Kumar et al. [196] focused on the architecture of computation offloading in MCC. Shi et al. [4] surveyed the work about MEC-based computation offloading. Unlike other survey papers, the authors focused on the different use cases, such as video analytics, smart home, smart city. They presented several challenges and opportunities for such use cases. Peng et al. [197] surveyed the work about MEC-based computation offloading from the perspective of service adoption and provision. Feng et al. [126] conducted a comprehensive survey on the application, offloading objectives, and offloading approaches of computation offloading in MEC. They further analyzed the current challenges and future direction from the perspectives of subtasks dependency and online task requests,

server selection, real-time environment perception, and security. Lin et al. [198] conducted a comprehensive survey for computation offloading with MEC in terms of application partitioning, task allocation, resource management, and distributed execution. Moreover, some researchers also conducted the survey for fog-based computation offloading [199], [200]. Combining different RPNs in computation offloading is also one of the focuses of many researchers, so, Wang et al. focused on the field of cloud-edge cooperative computation offloading systems, and categorized related papers from the perspective of task type, offloading decision, optimization objective, mobility, and the type of cooperation [35].

In addition to the literature review mentioned above, some other literature surveys focus on the optimization method used in computation offloading, and use the technology as the viewpoint. For example, Shakarami et al. [201] surveyed the literature which used the game-theory to optimize the computation offloading process for mobile edge computing. Wang et al. [3] did a comprehensive survey for the mobile network architecture and surveyed the mobile edge caching technology used in mobile edge computing. Shakarami et al. [202] reviewed the papers which focused on machine learning-based computation offloading. In their work, the researchers classified machine learning-based computation offloading into three types: supervised learning-based mechanisms, unsupervised learning-based mechanisms, and reinforcement learning-based mechanisms.

Some investigations are carried out based on optimization objectives. Wu [203] conducted computation offloading investigations from the perspective of the trade-off between energy consumption and response time. Energy saving is also a significant key performance indicator (KPI) of computation offloading, and it is also a key direction that many researchers pay attention to, so Cong et al. [133] surveyed the mobile edge computing from the view of hierarchical energy optimization.

The above survey papers related to computational offloading conducted the survey from various perspectives: some from an architectural standpoint, some based on application scenarios, some through optimization methods, and others focusing on optimization objectives (as shown in Table III.). However, they all overlook the impact of different types of tasks on computation offloading. This has prompted us to conduct a study based on task types.

IV. COMPUTATION OFFLOADING WITH STATIC TASK

The work related to the computation offloading of static tasks (Section IV) and the computation offloading of dynamic tasks (Section V) are surveyed separately. In addition to the task type, it can also be seen from step two in Fig. 3 that the offloading strategy also has a great influence on the algorithm for resource allocation and task allocation. Therefore, after classifying the computation offloading according to the task type, it further divides each category based on offloading strategy (i.e., full offloading and partial offloading).

In this section, the research on computation offloading involving static task are divided into two categories, full offloading (IV-A) and partial offloading (IV-B), according to offloading strategy, to conduct surveys separately.

TABLE III
RELATED SURVEY PAPERS

Main Survey Entry Point		Surveys
Scenario	IoT	[191]
	Vehicle	[192]
Type of RPN	MCC	[2], [193], [194], [196]
	MEC	[4], [126], [197], [198]
	Fog	[199], [200]
	Cloud-edge	[35]
Optimization Method	Game-theory	[201]
	Caching	[3]
	AI	[202]
Optimization Problem	Trade-off	[203]
	Energy	[133]

A. Full Offloading

1) Static Task With Energy Consumption Minimization Problem: Addressing the problem of minimizing energy consumption while meeting latency requirements in static task based computation offloading, Zhao et al. [81] studied it based on the static task in multi UE system. They jointly considered task allocation, radio resource allocation and computing resource allocation for the UE energy minimization problem. In order to solve the problem, the author proposed Reformulation-Linearization-Technique based Branch-and-Bound method.

Jointly considering resource allocation and task allocation can improve resource utilization to improve energy saving. Still, because the number of tasks for processing has not changed, the performance improvement is limited. Liu et al. [71] further considered edge caching in computation offloading, and they jointly optimized communication resource, computing resource, and caching contents to minimize the energy consumption of UE while satisfying the UE's latency requirement. To solve this optimization problem, which is proven as a mixed-integer non-convex optimization problem, the authors proposed an iterative algorithm based on the joint application of block coordinate descent and convex optimization.

Expanding on energy consumption considerations, in some studies, the energy consumption of RPN is also necessary to consider. The entire energy consumption minimization (UE and RPN) problem was considered in [76]. In their study, authors proposed a game theory-based approach for energy minimization problem. According to the rewards and punishments obtained after each iteration, iteratively update the offloading decision until the system reaches the Nash equilibrium.

2) Static Task With Latency Minimization Problem: The latency minimization problem was studied in the edge-could system by Wu et al. [141]. In their assumption, the cloud server can provide all services, but edge server can only provide part of services. They formulated the problem as an integer linear programming model, and they designed a hybrid heuristic method based on genetic algorithms and the simulated annealing selection strategy. Furthermore, regarding to the objective of minimizing latency in cloud-edge architecture, Ren et al. [82] further studied the latency minimization problem under assumptions that each user is associated with an edge server and all tasks have the same type and arrive at the same time. They decomposed the weighted sum latency minimization problem into two sub-problem: 1) minimizing the weighted transmission latency between UE and edge server; 2) minimizing the weighted computing latency of the edge server and the cloud server. For the first sub-problem, the Cauchy-Buniakowsky-Schwarz inequality can be used to obtain the optimal solution in closed form. For sub-problem two, the authors transformed it into a convex optimization problem. After that, Karush–Kuhn–Tucker (KKT) conditions can be used to solve the two sub-problems. Energy consumption was used as a constraint to participate in the optimization of computation offloading in [82] and [141].

3) Static Task With Trade-off Problem: Recognizing the significance of balancing energy consumption and latency, the trade-off between energy consumption and latency as a popular optimization objective was considered by Wang et al. [97]. They proposed a scheme by jointly considering the offloading strategy, interference management, and resource allocation in the computation offloading to minimize the weighted energy consumption and latency. Unlike some papers, the authors assumed that decision for resource and task allocation is determined by MEC server. The MEC server used the graph coloring method to allocate the offloading decision and physical resource block. Authors considered the same scheme in [101].

One challenge of computation offloading is communication. When the quality of the channel is poor (such as obstacle occlusion), the success rate of offloading will be greatly affected, especially in the case of full offloading. Designed for this condition, as a device deployed at high altitude, UAV can be used as a relay to solve this problem well. Moreover, UAV can be seen as a fog node that has a limited computing ability that can help UE to process some tasks [18], [131]. At the very beginning, UAV was regarded as a relaying node and existed in the computation offloading systems, and the computing ability is ignored by researchers [160], [204]. However, as research into computation offloading continues, the computing ability of UAV has also attracted the attention of researchers. Yu et al. [18] proposed a UAV-enabled MEC architecture to overcome the problem of the poor channel between IoT devices and a MEC server. They aimed to minimize the weighted latency and energy consumption (UE and UAV). In their consideration, the system consisted of a set of UEs (no computing ability), a UAV (low computing ability), and a set of ground edge servers (high computing ability). The authors proposed a successive convex approximation algorithm to find a sub-optimal offloading solution for minimizing the weighted sum of the latency and energy consumption of all UEs and UAV by jointly considering the UAV position, communication resource allocation and computing resource allocation.

Continuing the exploration of trade-off between energy consumption and latency, another study about the trade-off between energy consumption and latency was introduced in [99], the computation offloading worked on a multi-cell wireless network, and the authors' research purpose was minimizing the weighted sum of task completion time and energy consumption by jointly optimizing task allocation and resource allocation. Since this optimization problem is a mixed-integer nonlinear program, which is difficult to solve, the author decomposed this optimization problem into two sub-problems, including the resource allocation problem when the offloading decision is known, and the task allocation problem when the resource allocation is known. Finally, the optimal solution is approached by iteratively solving these two sub-problems and updating the known conditions.

It is also a trade-off between energy and latency issues, but Chen et al. [108] considered it in mobile edge-cloud computing. They studied the multi-UE computation offloading problem with multi-channels interference. For solving this trade-off problem, they used game theory (GT) to design a computation offloading model. Through continuous iterations, the system can reach the state of Nash equilibrium, and a suitable solution for offloading decision and resource allocation decision can be obtained. GT is used to model the interaction between two or more users, and it is a mathematical model [23], [176], [205]. UEs can effectively make decisions based on local observations by using GT in task and resource allocation. It has lower complexity, and it is often used in the multi-user computation offloading, but it often yields a suboptimal solution.

B. Partial Offloading

In terms of partial offloading with static task, most papers on computation offloading are based on multi-task (each UE has multi-tasks). In partial offloading, it is necessary to consider whether the UE needs to offload tasks and which tasks to offload. This is also the main reason why the algorithm of partial offloading is more complicated than the algorithm of full offloading.

1) Static Task With Energy Consumption Minimization Problem: Wu et al. [62] investigated the UE energy minimization problem while guaranteeing the latency requirement for the static task in multi-task scenarios. They achieved this by exploring computation offloading through non-orthogonal multiple access (NOMA) and jointly considering task allocation, local computing resource allocation, and NOMA transmission duration. The authors proposed an iterative optimization approach that optimizes task allocation (including the offloading decision and offloading order) and other resource allocation to achieve better performance.

2) Static Task With Latency Minimization Problem: With the development of the data-demanding application, the requirements for the communication of VR are also getting higher and higher. To support such applications, Du et al. [22] also investigated the use of THz wireless for MEC computation offloading systems in VR to minimize latency. Authors jointly took into account the viewport rendering and THz downlink power allocation problem by using the asynchronous advantage actor-critic algorithm, and a method based on deep reinforcement learning was proposed to learn the best viewport rendering position and transmission power control, and to adapt the time-varying characteristics of the wireless channel.

3) Static Task With Trade-off Problem: In [90], Wang et al. considered the case of multi-UE and multi-cloud computation offloading. Each UE has multiple tasks, and the weighted energy consumption and time consumption minimization problem can be formulated as an integer linear programming problem. The authors also optimized this problem separately for two different cases in their paper. For the special case, which is UE has unlimited energy and each task has the same resource requirements, the authors designed a polynomial-time optimal solution based on a weighted bipartite matching problem. The authors also proposed a novel heuristic-based algorithm to obtain the binary offloading decisions and the communication resource allocation method for the general case.

Similar to [90], Chen et al. [85] also considered the problem of multi-task in a multi-user system for the energy and latency trade-off problem. The optimization problem was modeled as an NP-hard, non-convex, quadratic constrained quadratic programming problem, and the authors proposed a separable semidefinite relaxation with the heuristic algorithm. Moreover, since the author considered the problem of multi-task, this will result in an overlap of task processing times and transmission times. Still, the authors did not analyze this overlap, they only considered the time lower bound (the degree of overlap in processing time reaches the maximum, only the largest one need to be taken into account) and time upper bound (no overlap in processing time), and use their corresponding performance (obtained by different method) as the lower and upper limits of system performance respectively for the performance benchmarking. Building upon [85], as a further extension, Chen et al. [94] further considered the multi-level collaborative computation offloading architecture in the multi-task multi-UE system for energy and latency trade-off problem. Based on the modeling in [85], the computing access point (a limited computing node located between UE and cloud) was taken into account. For solving this problem, the authors proposed a method called 'MUMTO-c', which has a similar principle with the method proposed in [85].

V. COMPUTATION OFFLOADING WITH DYNAMIC TASK

In this section, we will divide the computation offloading research involving dynamic task into two categories, full offloading and partial offloading, according to offloading strategy, to conduct surveys separately. The two types of dynamic task are surveyed in this section, i.e., naturally generated dynamic tasks (type one) and dynamic tasks generated by intervening with static tasks (type two).

A. Full Offloading

1) Type One of Dynamic Task: The energy consumption minimization while meeting the latency requirement problem was investigated in [60]. Zhang et al. modeled the MCC-based computation offloading with optimal-energy under the random

wireless channel. When the task is determined to be offloaded to the MCC server for processing, the data transmission rate is dynamically adjusted according to the current channel state information. When the task is determined for processing locally, the working frequency of the CPU is dynamically adjusted according to the immediate processing situation. In the study, the authors assumed that the input data size is known, but the CPU cycles requirement to process the corresponding task can be shown as a random number with an empirical distribution. Therefore, the processing frequency can be dynamically adjusted according to the distribution of random numbers and the actual processing conditions.

As a further extension of work at [60], You et al. [56] further considered the EH problem used in that case. In their study, they made the same assumptions for the task in this study as the task in [60]. The proposed microwave power transfer (MPT) and MCC combined based computation offloading for the passive low-complexity devices, and all the energy for mobile devices came from energy-harvesting. When the task was determined to offload to cloud for processing, it need divide the entire time into two time slots. The first time slot need to do MPT for energy collection, and did the computation offloading in time slot two by using the collected energy. When the task was determined to be processed locally, it will optimize the CPU frequency, the same idea as mentioned in [60]. Different from the considerations of many papers, in these two papers, they assumed that they do not know the detailed information of processing tasks such as the CPU cycles requirement of each task, and they assumed that the CPU cycles requirement is a random variable with an empirical distribution. The situation they considered is a computation offloading based on task unawareness, which can be defined as the dynamic task based computation offloading.

The studies conducted by [56] and [60] are categorized under a scenario where clock frequency is optimized continuously during the progression of computing. This is a type of optimization for computation offloading with dynamic task in the time domain. In addition, some researchers consider the impact of queue length in the buffer in previous time slot. Labidi et al. [74] thought that the number of data packets (task) arrival can be described by Poisson distribution. For the energy consumption minimization problem while meeting the latency requirement, Labidi proposed a deterministic and random offline strategy for a single UE system. The dynamic environment of the time-varying channel, wireless resource scheduling and computation load were jointly considered during the offloading process. As a further extension, [206] extended the single UE system of [74] to multi-UE system.

In the direction of energy saving for UE, in addition to latency can be used as a constraint, reliability can also be used as a constraint. Liu et al. [59] studied the UE energy consumption minimization problem of computation offloading in a multi-UE, multi-MEC, ultra-reliable low-latency edge computing scenario with the constrain of reliability and latency. By comparing with the current system designs which are relying on the average queue length, latency, the authors proposed an approach by imposing a probabilistic constraint on the queue length (probability of exceeding length) and

using extreme value theory to deal with extreme events. In that paper, task arrival will change over time which is the same as [56], but the authors considered modeling the task arrival by Poisson distribution. Considering that this way of task arrival will also cause queue congestion, this also motivated the author to consider the queue problem on the other hand. The queuing problem is also a common problem that often needs to be taken into account for dynamic tasks.

In [86], Liu et al. proposed a scheme to minimize latency. This was accomplished by finding the optimal offloading decision strategy based on the application buffer queue status, the available processing capacity at the UE and the MEC service, and the channel characteristics between the UE and the MEC server. The authors used Markov decision process to model this problem, and then proposed an efficient one-dimensional search algorithm to find the optimal task scheduling strategy.

2) Type Two of Dynamic Task: By far, the most considered computation offloading related research for type two of dynamic task is task splitting. In [45], Liu et al. studied the trade-off between the latency and communication reliability in MEC computation offloading for ultra-reliable low latency communication. The authors gave an assumption that the task can be split into multiple subtasks, the granularity of task splitting has arbitrary precision, and sequentially offload each subtask with the entire given channel bandwidth. The author designed three algorithms based on heuristic search, reformulation linearization technique, and semi-definite relaxation, respectively, and solved the problem by optimizing edge node candidate selection, offloading ranking, and task allocation.

Taking into account the division of tasks that cannot be arbitrarily split, it is crucial to consider task topology, which pertains to the relationships and dependencies among tasks within a system, when splitting tasks. This avoids random splits that could cause confusion or errors and ensures the resulting subdivided tasks maintain coherence and relevance within the broader context of the system. Zhao et al. [114] addressed the challenge of efficient task offloading in MEC, where tasks have specific service requirements and a dependent order of execution. The researchers emphasized the implications of constrained service caching at edge nodes on task offloading decisions, which can result in infeasible decisions or longer completion times. To address this issue, the authors defined the problem of offloading dependent tasks with service caching and proves that no constant approximation algorithm exists for this problem. The authors then proposed an efficient convex programming based algorithm to solve this problem and a favorite successor based algorithm to solve the special case with a homogeneous MEC.

Similarly, regarding the issue of task topology in computation offloading, Chen et al. [207] proposed a dependency-aware offloading scheme in MEC that utilizes both edge and remote cloud servers for latency minimization problem. The offloading problem is divided into two sub-problems (proved as NP-hard problems), each aiming to minimize the application finishing time under different cooperation modes and task dependency constraints. Then, two greedy-based algorithms were designed to solve the two sub-problems that were proven to be NP-hard. Simulation

results demonstrate that the proposed algorithms achieve near-optimal performance and outperform existing benchmark algorithms.

B. Partial Offloading

1) Type One of Dynamic Task: Liu and Zhang extended the full offloading method proposed in [59] to study partial offloading in their subsequent work [72]. The authors considered the same one-to-two task splitting as mentioned in [72], and they considered a resource allocation problem based on the ultra-reliable low-latency edge computing system for energy efficiency. In convention computation offloading, most of the computation offloading systems were designed based on average-based metrics, which is not suitable to be used in the ultra-reliable low-latency edge computing system (reliability requirement). Thus, Liu et al. proposed a new constraint design approach that is suitable for the ultra-reliable low-latency edge computing system by using the extreme value theory to offset the shortcomings of the design which is based on average-based metrics. In addition, the authors used the mobility characteristics of UE, and proposed a dual time scale UE-server association and task computation framework. In this regard, taking into account the task queue, the computing power, workload of the server, co-channel interference, and ultra-reliable and low-latency constraints, the authors used the matching theory to associate the UE with the MEC server for a long period of time. Then, given the associated MEC server, perform computation offloading and resource allocation in a short time.

Aside from task splitting, energy harvesting for computational offloading is also a widely researched area for type one dynamic tasks. The scheme proposed in [56] that all the energy of UE comes from the energy-harvesting, and it has great limitations. For example, the efficiency of EH is very low, so the energy it provided cannot support high-power computing and transmission, which is extremely unfriendly for some computation-intensive tasks. Therefore, in [105], the authors considered both the green power collected by EH and the energy of the battery of the mobile device itself. They also considered the situation where workload arrival will change over time, and then proposed an efficient resource management algorithm based on reinforcement learning. The algorithm instantly learns the best strategy for dynamic workload offloading and edge server configuration to reduce long-term system costs to the lowest. Foresight and adaptablity are supposed as two key points for the designed system.

2) Type Two of Dynamic Task: The problem of saving UE energy using EH and task splitting technology for type two dynamic tasks in computation offloading is investigated in [65]. Zhang et al. considered a combination of EH and task splitting. In their assumption, energy-constrained mobile devices harvest energy from ambient radio frequency signals, and the task can be split into two parts at any ratio (local processing part and MCC computing part). Then, jointly considering the clock frequency, transmission power and offload rate of UE to minimize the energy cost of UE by using the alternative optimization based on the difference between convex function programming and linear programming

Still in an arbitrary task splitting method for type two dynamic task, weighted UE energy consumption minimization problem in multi-UEs is considered in [107]. You et al. studied the multi-user mobile edge computing offloading system based on time-division multiple access (TDMA), and the task splitting was considered. Input data can be split into two part, one for local processing, one for remote processing. By using TDMA, it divide time into two time slots, one for transmission or local computing, the other time slot for cloud computing and downloading the task result. Moreover, assigning the offloading priority to the UE according to the status of the UE, if the UE has a lower priority, it offloads only a minimum amount of computation tasks to meet the latency requirement. Otherwise, it will offload all computation tasks to MEC. Resource utilization has been effectively improved by using the task splitting which changes the static task into dynamic task. Orthogonal frequency-division multiple access is also considered as extension work in [67], which has a get a better performance in energy saving, and the energy consumption is reduced by 90%.

Another code base task splitting for type two dynamic task is investigated in [106]. As a further extension of [45], Liu proposed a computation offloading based on code-partitioning in [106]. The authors still considered the trade-off problem between latency and reliability, but the difference is that, the reliability considered by the author was service reliability which was combined with communication reliability, computation-reliability, and the probability that the latency does not exceed the requirements. In order to solve this trade-off problem between reliability and latency, the author proposed an algorithm based on integer particle swarm optimization (IPSO). Although its result is close to the optimal solution of the problem, the complexity of the algorithm is too high. Therefore, the author proposed a heuristic algorithm with lower algorithm complexity but performance similar to IPSO.

Task splitting indeed provides a valuable way to transform static tasks into dynamic ones (type two), resulting in performance improvements. However, as mentioned, this approach does not optimize computation offloading from the perspective of input data, and a large amount of redundant data may still be used for data transmission and processing. Zhang et al. [57] recognized this issue, and they addressed this issue by jointly considering task allocation, resource allocation, and data removal in computation offloading systems to minimize UE energy consumption. They observed that in the time and task domains, input data corresponding to most tasks is redundant and detrimental to task processing. To tackle this problem, the authors proposed a similarity check method in the time domain and the task domain, aiming to remove highly correlated data, reduce redundancy, and improve the UE's energy efficiency. This approach marks a shift in the focus of computation offloading research, with Zhang et al. paying more attention to the correlation between source/data in the time domain. Similar views can be found in [30] and [168], further emphasizing the importance of addressing data redundancy and correlation in computation offloading systems.

VI. Conclusion and Future Computation Offloading

As shown in the previous sections, computation offloading has attracted significant attention in recent years. Researchers have studied computation offloading from various perspectives, and their results reflect the superiority of computation offloading in energy saving, latency reduction, and more. In this section, we will present our conclusion and summarize our understanding of future computation offloading and the corresponding challenges.

In recent years, there has been a significant increase in research related to computation offloading (see Fig. 1), as it has gained considerable attention and interest from the scientific community. According to our survey, these computational offloading studies predominantly concentrate on allocation steps, including resource allocation and task assignment. These aspects have been extensively researched and explored in the field for static task based computation offloading. In the field of dynamic task based computation offloading, type one dynamic tasks are often investigated in scenarios that involve ultra-reliable and low-latency computation offloading requirements, focusing on task queuing problems with extreme theory. Type two dynamic task are often investigated from the perspective of task splitting with optimum spitting ratio optimization problem. The splitting-ratio problem for different scenarios/optimization objectives are also well studied.

In computation offloading systems, energy efficiency can be represented by dividing the total payment by the total data size. Total payment may include factors such as time cost, energy cost, monetary cost, or others, depending on the optimization objective. In conventional computation offloading, the total data size of tasks is a fixed value that cannot be optimized. The only thing that can optimize is the payment by using resource allocation, task allocation. This also implies that there is a fixed minimum value for the total payment, which cannot be surpassed, regardless of employing techniques such as task splitting, game theory, machine learning, or heuristic algorithms. Consequently, when the total payment reaches its upper bound, conventional approaches become ineffective. This defines the upper bound of performance for conventional computation offloading.

As a result, the type two dynamic task with redundancy removing deserves to be further investigated for computation offloading. By focusing on this type of task, researchers can potentially uncover new ways to optimize the total data size and improve energy efficiency, latency reduction, and overall performance in computation offloading systems. Exploring redundancy removal methods, such as the similarity check method proposed by Zhang et al. [57], could lead to breakthroughs that significantly enhance the capabilities of computation offloading and expand its applications.

The aim of redundancy removing is to remove data that has no effect on task processing, this also involves the co-design of communication system and control signal (result of task). So, It is believed that the research on computation offloading will turn to computation offloading for communication and control co-design. In this regard, the next two points need to be studied:

- The modeling for redundancy removing based computation offloading under the stationary data.
- The modeling for redundancy removing based computation offloading under the non-stationary data.

The data can be divided into two different types, stationary data (the frequency of data change is predictable) and non-stationary data (the frequency of source change is unpredictable). For different types of data, their most suitable optimization methods are different. However, the current redundancy removing based work for computation offloading does not pay attention to this because they did not consider the complexity of the redundancy removing algorithm and its corresponding time and energy consumption. They only care about energy consumption and latency caused by communication and computing as well as they focus on the computation offloading from the task level. They do not make the most of the impact of the data in their work

Partial offloading will significantly increase the complexity of task allocation algorithms, so the corresponding algorithms (resource allocation and task allocation) will consume a lot of energy and cause high latency (although the time and energy cost from this part are ignored by most of the researchers now). The complexity problem is often overlooked in computation offloading, and this problem is challenging to solve. On the contrary, although full offloading has a lower flexibility perspective, its algorithmic complexity is also lower and it is easier to implement in industry. Therefore, full offloading will be the most appropriate offloading strategy when complexity issues are not addressed, because of the low algorithmic complexity of full offloading. Furthermore, different data collected by different sensors of the same UE, or data collected by different UEs, may be correlated. Using full offloading, different UEs can share the data at the RPN to achieve higher reliability than using only the data collected by the UE itself, or reduce the energy consumption and latency caused repeated computation. Moreover, when moving to 6G, THz communication becomes more and more critical. So that how to combine the full offloading with THz communication will be a hot spot research direction. In particular, the integrated and sensing communication [208], [209] under THz communication has the characteristics that sensing and communication can be carried out at the same time, which is very conducive to full offloading and data removing, but the work in this direction has not yet started.

References

[1] H. J. La and S. D. Kim, "A taxonomy of offloading in mobile cloud computing," in *Proc. IEEE 7th Int. Conf. Service-Oriented Comput. Appl.*, Nov. 2014, pp. 147–153.

[2] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[3] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[5] R. P. Mathur and M. Sharma, "A survey on computational offloading in mobile cloud computing," in *Proc. 5th Int. Conf. Image Inf. Process. (ICIIP)*, Nov. 2019, pp. 515–520.

[6] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward computation offloading in edge computing: A survey," *IEEE Access*, vol. 7, pp. 131543–131558, 2019.

[7] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial Internet of Things: Architecture, advances and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2462–2488, 4th Quart., 2020.

[8] S. M. A. Kazmi et al., "A novel contract theory-based incentive mechanism for cooperative task-offloading in electrical vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8380–8395, Jul. 2022.

[9] K. Xiong, S. Leng, C. Huang, C. Yuen, and Y. L. Guan, "Intelligent task offloading for heterogeneous V2X communications," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2226–2238, Apr. 2021.

[10] X. Xu, X. Zhang, X. Liu, J. Jiang, L. Qi, and M. Z. A. Bhuiyan, "Adaptive computation offloading with edge for 5G-envisioned Internet of Connected vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5213–5222, Aug. 2021.

[11] Z. Ning et al., "Intelligent edge computing in Internet of Vehicles: A joint computation offloading and caching solution," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2212–2225, Apr. 2021.

[12] Y. Liu et al., "Physical layer security assisted computation offloading in intelligently connected vehicle networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3555–3570, Jun. 2021.

[13] Q. Luo, C. Li, T. H. Luan, W. Shi, and W. Wu, "Self-learning based computation offloading for Internet of Vehicles: Model and algorithm," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 5913–5925, Sep. 2021.

[14] Y. Wu, J. Wu, L. Chen, J. Yan, and Y. Han, "Load balance guaranteed vehicle-to-vehicle computation offloading for min-max fairness in VANETs," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11994–12013, Aug. 2022.

[15] M. Messous, S. Senouci, H. Sedjelmaci, and S. Cherkaoui, "A game theory based efficient computation offloading in an UAV network," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4964–4974, May 2019.

[16] X. Cao, J. Xu, and R. Zhang, "Mobile edge computing for cellular-connected UAV: Computation offloading and trajectory optimization," in *Proc. IEEE Int. Workshop Signal Process. Adv. Wireless Commu. (SPAWC)*, Oct. 2018, pp. 1–5.

[17] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint task offloading and resource allocation in UAV-enabled mobile edge computing," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3147–3159, Apr. 2020.

[18] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint task offloading and resource allocation in UAV-enabled mobile edge computing," *IEEE Trans. Cybern.*, vol. 7, no. 4, pp. 3147–3159, Sep. 2020.

[19] M. Dai, Z. Su, Q. Xu, and N. Zhang, "Vehicle assisted computing offloading for unmanned aerial vehicles in smart city," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1932–1944, Mar. 2021.

[20] M. Chen, W. Saad, C. Yin, and M. Debbah, "Data correlation-aware resource management in wireless virtual reality (VR): An echo state transfer learning approach," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4267–4280, Jun. 2019.

[21] M. Chen, W. Saad, and C. Yin, "Liquid state based transfer learning for 360° image transmission in wireless VR networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Aug. 2019, pp. 1–6.

[22] J. Du, F. R. Yu, G. Lu, J. Wang, J. Jiang, and X. Chu, "MEC-assisted immersive VR video streaming over terahertz wireless networks: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9517–9529, Oct. 2020.

[23] Z. Hong, H. Huang, S. Guo, W. Chen, and Z. Zheng, "QoS-aware cooperative computation offloading for robot swarms in cloud robotics," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 4027–4041, Apr. 2019.

[24] Y. Zhai et al., "Cooperative offloading for multiple robot applications," in *Proc. IEEE Int. Conf. Joint Cloud Comput.*, Aug. 2020, pp. 63–70.

[25] X. Wang and H. Guo, "Mobility-aware computation offloading for swarm robotics using deep reinforcement learning," in *Proc. IEEE 18th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2021, pp. 1–4.

[26] S. Gupta and J. Chakareski, "Lifetime maximization in mobile edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3310–3321, Mar. 2020.

[27] Y. Shen and B. Guo, "Energy-efficient cluster-head selection with fuzzy logic for robotic fish swarm," in *Proc. 12th Int. Conf. Fuzzy Syst. Knowl. Discovery (FSKD)*, Aug. 2015, pp. 513–518.

[28] Y. Guo, Z. Mi, Y. Yang, and M. S. Obaidat, "An energy sensitive computation offloading strategy in cloud robotic network based on GA," *IEEE Syst. J.*, vol. 13, no. 3, pp. 3513–3523, Sep. 2019.

[29] S. Zhang, N. Yi, and Y. Ma, "Robot subset selection for swarm lifetime maximization in computation offloading with correlated data sources," in *Proc. ICC IEEE Int. Conf. Commun.*, May 2023, pp. 1–20.

[30] B. Nour and S. Cherkaoui, "A network-based compute reuse architecture for IoT applications," 2021, *arXiv:2104.03818*.

[31] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013.

[32] Q.-H. Nguyen and F. Dressler, "A smartphone perspective on computation offloading—A survey," *Comput. Commun.*, vol. 159, pp. 133–154, Jun. 2020.

[33] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[34] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. MCC Workshop Mobile Cloud Comput.*, Aug. 2012, pp. 13–16.

[35] B. Wang, C. Wang, W. Huang, Y. Song, and X. Qin, "A survey and taxonomy on task offloading for edge-cloud computing," *IEEE Access*, vol. 8, pp. 186080–186101, 2020.

[36] S. D. Okegbile, B. T. Maharaj, and A. S. Alfa, "A multi-user tasks offloading scheme for integrated edge-fog-cloud computing environments," *IEEE Trans. Veh. Technol.*, vol. 71, no. 7, pp. 7487–7502, Jul. 2022, doi: 10.1109/TVT.2022.3167892.

[37] K. Cao, S. Hu, Y. Shi, A. W. Colombo, S. Karnouskos, and X. Li, "A survey on edge and edge-cloud computing assisted cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7806–7819, Nov. 2021.

[38] E. Biglieri, J. Proakis, and S. Shamai (Shitz), "Fading channels: Information-theoretic and communications aspects," *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2619–2692, Oct. 1998.

[39] G. Wang, X. Yu, F. Xu, and J. Cai, "Task offloading and resource allocation for UAV-assisted mobile edge computing with imperfect channel estimation over rician fading channels," *EURASIP J. Wireless Commun. Netw.*, vol. 2020, no. 1, pp. 1–19, Sep. 2020.

[40] B. Shang, L. Liu, and Z. Tian, "Deep learning-assisted energy-efficient task offloading in vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9619–9624, Sep. 2021.

[41] J. Xue and Y. An, "Joint task offloading and resource allocation for multi-task multi-server NOMA-MEC networks," *IEEE Access*, vol. 9, pp. 16152–16163, 2021.

[42] X. Liu, J. Liu, and H. Wu, "Energy-efficient task allocation of heterogeneous resources in mobile edge computing," *IEEE Access*, vol. 9, pp. 119700–119711, 2021.

[43] M. Huang, W. Liu, T. Wang, A. Liu, and S. Zhang, "A cloud–MEC collaborative task offloading scheme with service orchestration," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5792–5805, Jul. 2020.

[44] M. Qin et al., "Service-oriented energy-latency tradeoff for IoT task partial offloading in MEC-enhanced multi-RAT networks," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1896–1907, Feb. 2021.

[45] J. Liu and Q. Zhang, "Offloading schemes in mobile edge computing for ultra-reliable low latency communications," *IEEE Access*, vol. 6, pp. 12825–12837, 2018.

[46] Q. Ye, W. Shi, K. Qu, H. He, W. Zhuang, and X. Shen, "Joint RAN slicing and computation offloading for autonomous vehicular networks: A learning-assisted hierarchical approach," *IEEE Open J. Veh. Technol.*, vol. 2, pp. 272–288, 2021.

[47] H. Hu, Q. Wang, R. Q. Hu, and H. Zhu, "Mobility-aware offloading and resource allocation in a MEC-enabled IoT network with energy harvesting," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17541–17556, Dec. 2021.

[48] P. A. Apostolopoulos, G. Fragkos, E. E. Tsiropoulou, and S. Papavassiliou, "Data offloading in UAV-assisted multi-access edge computing systems under resource uncertainty," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 175–190, Jan. 2023.

[49] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digit. Commun. Netw.*, vol. 5, no. 1, pp. 10–17, Feb. 2019.

[50] C. Li, M. Song, H. Tang, and Y. Luo, "Offloading and system resource allocation optimization in TDMA based wireless powered mobile edge computing," *J. Syst. Archit.*, vol. 98, pp. 221–230, Sep. 2019.

[51] L. Liu, M. Zhao, M. Yu, M. A. Jan, D. Lan, and A. Taherkordi, "Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 2169–2182, Feb. 2023.

[52] L. Zhang, M. Jia, J. Wu, Q. Guo, and X. Gu, "Joint task secure offloading and resource allocation for multi-MEC server to improve user QoE," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Jul. 2021, pp. 103–108.

[53] J. Wang, Y. Ma, N. Yi, and R. Tafazolli, "On URLLC downlink transmission modes for MEC task offloading," in *Proc. IEEE 91st Veh. Technol. Conf.*, May 2020, pp. 1–5.

[54] L. Wang, X. Sun, R. Jiang, W. Jiang, Z. Zhong, and D. W. K. Ng, "Optimal energy efficiency for multi-MEC and blockchain empowered IoT: A deep learning approach," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2021, pp. 1–6.

[55] X. Shan, H. Zhi, P. Li, and Z. Han, "A survey on computation offloading for mobile edge computing information," in *Proc. IEEE IEEE 4th Int. Conf. Big Data Secur. Cloud (BigDataSecurity) Int. Conf. High Perform. Smart Comput., (HPSC) IEEE Int. Conf. Intell. Data Secur. (IDS)*, May 2018, pp. 248–251.

[56] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1757–1771, May 2016.

[57] S. Zhang, N. Yi, and Y. Ma, "Correlation-based device energy-efficient dynamic multi-task offloading for mobile edge computing," in *Proc. IEEE 93rd Veh. Technol. Conf. (VTC-Spring)*, Apr. 2021, pp. 1–5.

[58] O. Mu noz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.

[59] C.-F. Liu, M. Bennis, and H. V. Poor, "Latency and reliability-aware task offloading and resource allocation for mobile edge computing," in *Proc. IEEE Globecom Workshops*, Dec. 2017, pp. 1–7.

[60] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[61] W. Zhang and Y. Wen, "Cloud-assisted collaborative execution for mobile applications with general task topology," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 6815–6821.

[62] Y. Wu, B. Shi, L. P. Qian, F. Hou, J. Cai, and X. S. Shen, "Energy-efficient multi-task multi-access computation offloading via NOMA transmission for IoTs," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4811–4822, Jul. 2020.

[63] J. H. Anajemba, T. Yue, C. Iwendi, M. Alenezi, and M. Mittal, "Optimal cooperative offloading scheme for energy efficient multi-access edge computation," *IEEE Access*, vol. 8, pp. 53931–53941, 2020, doi: 10.1109/ACCESS.2020.2980196.

[64] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 301–313, Apr. 2019.

[65] Y. Zhang, J. He, and S. Guo, "Energy-efficient dynamic task offloading for energy harvesting mobile cloud computing," in *Proc. IEEE Int. Conf. Netw., Archit. Storage (NAS)*, Chongqing, China, Oct. 2018, pp. 1–4.

[66] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.

[67] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[68] Y. Hao, M. Chen, L. Hu, M. S. Hossain, and A. Ghoneim, "Energy efficient task caching and offloading for mobile edge computing," *IEEE Access*, vol. 6, pp. 11365–11373, 2018.

[69] C. Liu and M. Bennis, "Ultra-reliable and low-latency vehicular transmission: An extreme value theory approach," *IEEE Commun. Lett.*, vol. 22, no. 6, pp. 1292–1295, Jun. 2018.

[70] C. You, Y. Zeng, R. Zhang, and K. Huang, "Asynchronous mobile-edge computation offloading: Energy-efficient resource management," *IEEE Trans. Wireless Commun.*, vol. 17, no. 11, pp. 7590–7605, Nov. 2018.

[71] P. Liu, G. Xu, K. Yang, K. Wang, and X. Meng, "Jointly optimized energy-minimal resource allocation in cache-enhanced mobile edge computing systems," *IEEE Access*, vol. 7, pp. 3336–3347, 2019.

[72] C. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4132–4150, Jun. 2019.

[73] Z. Zhou et al., "Learning-based URLLC-aware task offloading for Internet of Health Things," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 396–410, Feb. 2021.

[74] W. Labidi, M. Sarkiss, and M. Kamoun, "Energy-optimal resource scheduling and computation offloading in small cell networks," in *Proc. 22nd Int. Conf. Telecommun. (ICT)*, Apr. 2015, pp. 313–318.

[75] K. Zhang et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[76] Y. Ge, Y. Zhang, Q. Qiu, and Y.-H. Lu, "A game theoretic resource allocation for overall energy minimization in mobile cloud computing system," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, Jul. 2012, pp. 279–284.

[77] X. Cao, F. Wang, J. Xu, R. Zhang, and S. Cui, "Joint computation and communication cooperation for mobile edge computing," in *Proc. 16th Int. Symp. Model. Optim. Mobile, Ad Hoc, Wireless Netw.*, 2018, pp. 1–6.

[78] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3774–3785, Mar. 2021.

[79] M. Daraghmeh, I. Al Ridhawi, M. Aloqaily, Y. Jararweh, and A. Agarwal, "A power management approach to reduce energy consumption for edge computing servers," in *Proc. Int. Conf. Fog Mobile Edge Comput. (FMEC)*, 2019, pp. 259–264.

[80] J. Wang, W. Wu, Z. Liao, A. K. Sangaiah, and R. Simon Sherratt, "An energy-efficient off-loading scheme for low latency in collaborative edge computing," *IEEE Access*, vol. 7, pp. 149182–149190, 2019.

[81] P. Zhao, H. Tian, C. Qin, and G. Nie, "Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing," *IEEE Access*, vol. 5, pp. 11255–11268, 2017.

[82] J. Ren et al., "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[83] D. Nowak, T. Mahn, H. Al-Shatri, A. Schwartz, and A. Klein, "A generalized Nash game for mobile edge computation offloading," in *Proc. 6th IEEE Int. Conf. Mobile Cloud Comput., Services, Eng.*, Bamberg, Germany, Mar. 2018, pp. 95–102.

[84] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

[85] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2016, pp. 1–6.

[86] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2016, pp. 1451–1455.

[87] Y. Duan, C. She, G. Zhao, and T. Q. S. Quek, "Delay analysis and computing offloading of URLLC in mobile edge computing systems," in *Proc. 10th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2018, pp. 1–6.

[88] Z. Liu, Y. Yang, K. Wang, Z. Shao, and J. Zhang, "POST: Parallel offloading of splittable tasks in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3170–3183, Apr. 2020.

[89] Z. Bellal, B. Nour, and S. Mastorakis, "CoxNet: A computation reuse architecture at the edge," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 2, pp. 765–777, Jun. 2021.

[90] X. Wang, J. Wang, X. Wang, and X. Chen, "Energy and delay tradeoff for application offloading in mobile cloud computing," *IEEE Syst. J.*, vol. 11, no. 2, pp. 858–867, Jun. 2017.

[91] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9, doi: 10.1109/INFOCOM.2016.7524497.

[92] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[93] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing," *IEEE Access*, vol. 6, pp. 19324–19337, 2018.

[94] M.-H. Chen, B. Liang, and M. Dong, "Multi-user multi-task offloading and resource allocation in mobile cloud systems," *IEEE Wireless Commun.*, vol. 17, no. 10, pp. 6790–6805, Oct. 2018.

[95] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial IoT–Edge–Cloud computing environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2759–2774, Dec. 2019.

[96] S. Liang, H. Wan, T. Qin, J. Li, and W. Chen, "Multi-user computation offloading for mobile edge computing: A deep reinforcement learning and game theory approach," in *Proc. IEEE 20th Int. Conf. Commun. Technol. (ICCT)*, Oct. 2020, pp. 1534–1539.

[97] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7432–7445, Aug. 2017.

[98] J. Zhang et al., "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2633–2645, Aug. 2018.

[99] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[100] W.-J. Feng, C.-H. Yang, and X.-S. Zhou, "Multi-user and multi-task offloading decision algorithms based on imbalanced edge cloud," *IEEE Access*, vol. 7, pp. 95970–95977, 2019.

[101] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Joint computation offloading, resource allocation and content caching in cellular networks with mobile edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6.

[102] X. Ma, C. Lin, X. Xiang, and C. Chen, "Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing," in *Proc. 18th ACM Int. Conf. Model., Anal. Simul. Wireless Mobile Syst.*, Nov. 2015, pp. 271–278, doi: 10.1145/2811587.2811598.

[103] Y. Zhang, D. Niyato, and P. Wang, "Offloading in mobile cloudlet systems with intermittent connectivity," *IEEE Trans. Mobile Comput.*, vol. 14, no. 12, pp. 2516–2529, Dec. 2015.

[104] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[105] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Trans. Cognit. Commun. Netw.*, vol. 3, no. 3, pp. 361–373, Sep. 2017.

[106] J. Liu and Q. Zhang, "Code-partitioning offloading schemes in mobile edge computing for augmented reality," *IEEE Access*, vol. 7, pp. 11222–11236, 2019.

[107] C. You and K. Huang, "Multiuser resource allocation for mobile-edge computation offloading," in *Proc. IEEE Global Commun. Conf.*, Dec. 2016, pp. 1–6.

[108] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[109] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

[110] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, "Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16067–16081, Dec. 2020.

[111] T. Liu, J. Li, F. Shu, and Z. Han, "Optimal task allocation in vehicular fog networks requiring URLLC: An energy-aware perspective," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 3, pp. 1879–1890, Jul. 2020.

[112] Q. Wu, H. Ge, H. Liu, Q. Fan, Z. Li, and Z. Wang, "A task offloading scheme in vehicular fog and cloud computing system," *IEEE Access*, vol. 8, pp. 1173–1184, 2020.

[113] Z. Liu, P. Dai, H. Xing, Z. Yu, and W. Zhang, "A distributed algorithm for task offloading in vehicular networks with hybrid fog/cloud computing," *IEEE Trans. Syst. Man, Cybern., Syst.*, vol. 52, no. 7, pp. 4388–4401, Jul. 2022.

[114] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang, "Offloading tasks with dependency and service caching in mobile edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 11, pp. 2777–2792, Nov. 2021.

[115] R. Sheldon and S. J. Bigelow. What is Task Manager? TechTarget. Accessed: Jun. 6, 2024. [Online]. Available: https://www.techtarget.com/searchenterprisedesktop/definition/Microsoft-Windows-Task-Manager

[116] L. Ismail, H. Materwala, and H. S. Hassanein, "QoS-SLA-Aware adaptive genetic algorithm for multi-request offloading in integrated edge-cloud computing in Internet of Vehicles," 2022, *arXiv:2202.01696*.

[117] H. Materwala, L. Ismail, R. M. Shubair, and R. Buyya, "Energy-sla-aware genetic algorithm for edge–cloud integrated computation offloading in vehicular networks," *Future Gener. Comp. Syst.*, vol. 135, pp. 205–222, Oct. 2022.

[118] L. Ismail and H. Materwala, "ESCOVE: Energy-SLA-aware edge-cloud computation offloading in vehicular networks," *Sensors*, vol. 21, no. 15, p. 5233, Aug. 2021.

[119] L. Ismail and H. Materwala, "IoT-edge-cloud computing framework for QoS-aware computation offloading in autonomous mobile agents: Modeling and simulation," in *Mobile, Secure, and Programmable Networking*. Cham, Switzerland: Springer, 2021, pp. 161–176.

[120] C. Shang, Y. Sun, H. Luo, and M. Guizani, "Computation offloading and resource allocation in NOMA-MEC: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 10, no. 17, pp. 15464–15476, Sep. 2023.

[121] Y. Cong, K. Xue, C. Wang, W. Sun, S. Sun, and F. Hu, "Latency-energy joint optimization for task offloading and resource allocation in MEC-assisted vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 1, no. 1, pp. 11–13, Jun. 2023.

[122] K. Li, X. Wang, Q. He, Q. Ni, M. Yang, and S. Dustdar, "Computation offloading for tasks with bound constraints in multiaccess edge computing," *IEEE Internet Things J.*, vol. 10, no. 17, pp. 15526–15536, Sep. 2023.

[123] H. Yang et al., "Intelligent computation offloading for joint communication and sensing-based vehicular networks," *IEEE Trans. Wireless Commun.*, vol. 23, no. 4, pp. 3600–3616, Apr. 2024.

[124] Z. Ji, S. Wu, and C. Jiang, "Cooperative multi-agent deep reinforcement learning for computation offloading in digital twin satellite edge networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3414–3429, Nov. 2023.

[125] G. Calice, A. Mtibaa, R. Beraldi, and H. Alnuweiri, "Mobile-to-mobile opportunistic task splitting and offloading," in *Proc. IEEE 11th Int. Conf. Wireless Mobile Comput., Netw. Commun.*, Oct. 2015, pp. 565–572.

[126] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, and L. Guo, "Computation offloading in mobile edge computing networks: A survey," *J. Netw. Comput. Appl.*, vol. 202, Jun. 2022, Art. no. 103366.

[127] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4514–4526, May 2018.

[128] T. T. Nguyen, V. N. Ha, L. B. Le, and R. Schober, "Joint data compression and computation offloading in hierarchical fog-cloud systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 293–309, Jan. 2020.

[129] S. Li and J. Huang, "Energy efficient resource management and task scheduling for IoT services in edge computing paradigm," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Appl. IEEE Int. Conf. Ubiquitous Comput. Commun. (ISPA/IUCC)*, Dec. 2017, pp. 846–851.

[130] L. Feng et al., "Energy-efficient offloading for mission-critical IoT services using EVT-embedded intelligent learning," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 3, pp. 1179–1190, Sep. 2021.

[131] Z. Chen, N. Xiao, and D. Han, "A multilevel mobile fog computing offloading model based on UAV-assisted and heterogeneous network," *Wireless Commun. Mobile Comput.*, vol. 2020, pp. 1–11, Jul. 2020.

[132] Y. Sun, S. Zhou, and J. Xu, "EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 1, pp. 2637–2646, Nov. 2017.

[133] P. Cong, J. Zhou, L. Li, K. Cao, T. Wei, and K. Li, "A survey of hierarchical energy optimization for mobile edge computing: A perspective from end devices to the cloud," *ACM Comput. Surveys.*, vol. 53, no. 2, pp. 1–44, 2020.

[134] X. Xiang, C. Lin, and X. Chen, "Energy-efficient link selection and transmission scheduling in mobile cloud computing," *IEEE Wireless Commun. Lett.*, vol. 3, no. 2, pp. 153–156, Apr. 2014.

[135] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *Proc. IEEE Conf. Comput. Commun. Workshops*, Apr. 2014, pp. 352–357.

[136] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3056–3069, Nov. 2017.

[137] F. Sun et al., "Cooperative task scheduling for computation offloading in vehicular cloud," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11049–11061, Nov. 2018.

[138] C. She, Y. Duan, G. Zhao, T. Q. S. Quek, Y. Li, and B. Vucetic, "Cross-layer design for mission-critical IoT in mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 9360–9374, Dec. 2019.

[139] Q.-u.-A. Mastoi, A. Lakhan, F. A. Khan, and Q. H. Abbasi, "Dynamic content and failure aware task offloading in heterogeneous mobile cloud networks," in *Proc. Int. Conf. Adv. Emerg. Comput. Technol. (AECT)*, Feb. 2020, pp. 1–6.

[140] K. Guo and T. Q. S. Quek, "On the asynchrony of computation offloading in multi-user MEC systems," *IEEE Trans. Commun.*, vol. 68, no. 12, pp. 7746–7761, Dec. 2020.

[141] H. Wu, S. Deng, W. Li, S. U. Khan, J. Yin, and A. Y. Zomaya, "Request dispatching for minimizing service response time in edge cloud systems," in *Proc. 27th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Hangzhou, China, Jul. 2018, pp. 1–9.

[142] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing IoT service delay via fog offloading," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 998–1010, Apr. 2018.

[143] J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5506–5519, Aug. 2018.

[144] K. Sadatdiynov, L. Cui, L. Zhang, J. Z. Huang, S. Salloum, and M. S. Mahmud, "A review of optimization methods for computation offloading in edge computing networks," *Digit. Commun. Netw.*, vol. 9, no. 2, pp. 450–461, Apr. 2023.

[145] Y. Deng, Z. Chen, X. Chen, and Y. Fang, "Task offloading in multi-hop relay-aided multi-access edge computing," *IEEE Trans. Veh. Technol.*, vol. 72, no. 1, pp. 1372–1376, Jan. 2023.

[146] J. Chen, Y. Yang, C. Wang, H. Zhang, C. Qiu, and X. Wang, "Multitask offloading strategy optimization based on directed acyclic graphs for edge computing," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9367–9378, Jun. 2022.

[147] S. Liu, P. Cheng, Z. Chen, W. Xiang, B. Vucetic, and Y. Li, "User-oriented task offloading for mobile edge computing in ultra-dense networks," in *Proc. IEEE Global Commun. Conf.*, Dec. 2021, pp. 1–6.

[148] P. Cai, F. Yang, J. Wang, X. Wu, Y. Yang, and X. Luo, "JOTE: Joint offloading of tasks and energy in fog-enabled IoT networks," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3067–3082, Apr. 2020.

[149] X. Deng, Z. Sun, D. Li, J. Luo, and S. Wan, "User-centric computation offloading for edge computing," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12559–12568, Aug. 2021.

[150] S. Tong, Y. Liu, J. Mišic, X. Chang, Z. Zhang, and C. Wang, "Joint task offloading and resource allocation for fog-based intelligent transportation systems: A UAV-enabled multi-hop collaboration paradigm," *IEEE Trans. Intell. Transp. Syst.*, vol. 2, no. 1, pp. 1–16, Apr. 2022.

[151] H. Guo, J. Liu, and H. Qin, "Collaborative mobile edge computation offloading for IoT over fiber-wireless networks," *IEEE Netw.*, vol. 32, no. 1, pp. 66–71, Jan. 2018.

[152] J. Liu, J. Ren, Y. Zhang, X. Peng, Y. Zhang, and Y. Yang, "Efficient dependent task offloading for multiple applications in MEC-cloud system," *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2147–2162, Apr. 2023.

[153] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[154] L. Qiu, W.-J. Hsu, S.-Y. Huang, and H. Wang, "Scheduling and routing algorithms for AGVs: A survey," *Int. J. Prod. Res.*, vol. 40, no. 3, pp. 745–760, Jan. 2002.

[155] P. Pace, G. Aloi, R. Gravina, G. Caliciuri, G. Fortino, and A. Liotta, "An edge-based architecture to support efficient applications for healthcare Industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 15, no. 1, pp. 481–489, Jan. 2019.

[156] H. Al-Shatri, S. Müller, and A. Klein, "Distributed algorithm for energy efficient multi-hop computation offloading," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–6.

[157] C. Funai, C. Tapparello, and W. Heinzelman, "Computational offloading for energy constrained devices in multi-hop cooperative networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 1, pp. 60–73, Jan. 2020.

[158] D. Xu et al., "A survey of opportunistic offloading," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2198–2236, 3rd Quart., 2018.

[159] Y. Sahni, J. Cao, L. Yang, and Y. Ji, "Multi-hop multi-task partial computation offloading in collaborative edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1133–1145, May 2021.

[160] Y. Zeng, R. Zhang, and T. J. Lim, "Throughput maximization for UAV-enabled mobile relaying systems," *IEEE Trans. Commun.*, vol. 64, no. 12, pp. 4983–4996, Dec. 2016.

[161] S. Müller, H. Al-Shatri, M. Wichtlhuber, D. Hausheer, and A. Klein, "Computation offloading in wireless multi-hop networks: Energy minimization via multi-dimensional knapsack problem," in *Proc. IEEE 26th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Aug. 2015, pp. 1717–1722.

[162] Y. Zhao, S. Zhou, T. Zhao, and Z. Niu, "Energy-efficient task offloading for multiuser mobile cloud computing," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Shenzhen, China, Nov. 2015, pp. 1–5.

[163] J. Liu and Q. Zhang, "Reliability and latency aware code-partitioning offloading in mobile edge computing," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Jan. 2019, pp. 1–7.

[164] H. Flores, "Mobile code offloading: From concept to practice and beyond," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 80–88, Mar. 2015.

[165] S. Wang and S. Dey, "Adaptive mobile cloud computing to enable rich mobile multimedia applications," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 870–883, Jun. 2013.

[166] M. Molina, O. Mu noz, A. Pascual-Iserte, and J. Vidal, "Joint scheduling of communication and computation resources in multiuser wireless application offloading," in *Proc. IEEE 25th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, May 2014, pp. 1093–1098.

[167] M. Deng, H. Tian, and B. Fan, "Fine-granularity based application offloading policy in cloud-enhanced small cell networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC)*, Kuala Lumpur, Malaysia, May 2016, pp. 638–643.

[168] B. Nour and S. Cherkaoui, "How far can we go in compute-less networking: Computation correctness and accuracy," 2021, *arXiv:2103.15924*.

[169] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[170] Y. Zhang, P. Du, J. Wang, T. Ba, R. Ding, and N. Xin, "Resource scheduling for delay minimization in multi-server cellular edge computing systems," *IEEE Access*, vol. 7, pp. 86265–86273, 2019.

[171] M. S. Elbamby et al., "Wireless edge computing with latency and reliability guarantees," *Proc. IEEE*, vol. 107, no. 8, pp. 1717–1737, Aug. 2019.

[172] N. Kherraf, S. Sharafeddine, C. M. Assi, and A. Ghrayeb, "Latency and reliability-aware workload assignment in IoT networks with mobile edge clouds," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1435–1449, Dec. 2019.

[173] Y. K. Tun et al., "Energy efficient communication and computation resource slicing for eMBB and URLLC coexistence in 5G and beyond," *IEEE Access*, vol. 8, pp. 136024–136035, 2020.

[174] H. Guo, J. Liu, and J. Lv, "Toward intelligent task offloading at the edge," *IEEE Netw.*, vol. 34, no. 2, pp. 128–134, Mar. 2020.

[175] B. Yang, X. Cao, J. Bassey, X. Li, and L. Qian, "Computation offloading in multi-access edge computing: A multi-task learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 9, pp. 2745–2762, Sep. 2021.

[176] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan. 2018.

[177] S. Ranadheera, S. Maghsudi, and E. Hossain, "Mobile edge computation offloading using game theory and reinforcement learning," 2017, *arXiv:1711.09012*.

[178] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, and N. Georgalas, "Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 64–69, May 2019.

[179] S. Pan, Z. Zhang, Z. Zhang, and D. Zeng, "Dependency-aware computation offloading in mobile edge computing: A reinforcement learning approach," *IEEE Access*, vol. 7, pp. 134742–134753, 2019.

[180] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.

[181] Z. Ning, P. Dong, X. Wang, J. J. P. C. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, pp. 1–24, Oct. 2019, doi: 10.1145/3317572.

[182] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 242–253, Jan. 2021.

[183] D. K. Mishra and V. Shinde, "A review of global optimization problems using meta-heuristic algorithm," in *Nature-Inspired Optimization Algorithms: Recent Advances in Natural Computing and Biomedical Applications*. Berlin, Germany: De Gruyter, 2021, pp. 87–106, doi: 10.1515/9783110676112-006.

[184] Y. Sun, C. Song, S. Yu, Y. Liu, H. Pan, and P. Zeng, "Energy-efficient task offloading based on differential evolution in edge computing system with energy harvesting," *IEEE Access*, vol. 9, pp. 16383–16391, 2021.

[185] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.

[186] A. Rahman, J. Jin, A. Rahman, T. Cricenti, and A. Kulkarni, "Communication-aware cloud robotic task offloading with on-demand mobility for smart factory maintenance," *IEEE Trans. Ind. Informat.*, vol. 15, no. 5, pp. 2500–2511, May 2019.

[187] C. Wang, Y. Li, and D. Jin, "Mobility-assisted opportunistic computation offloading," *IEEE Commun. Lett.*, vol. 18, no. 10, pp. 1779–1782, Oct. 2014.

[188] D. Wang, Z. Liu, X. Wang, and Y. Lan, "Mobility-aware task offloading and migration schemes in fog computing networks," *IEEE Access*, vol. 7, pp. 43356–43368, 2019.

[189] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 3, pp. 443–461, 3rd Quart., 2011.

[190] A. Rahman, J. Jin, A. Cricenti, A. Rahman, and M. Panda, "Motion and connectivity aware offloading in cloud robotics via genetic algorithm," in *Proc. IEEE Glob. Commun. Conf.*, Aug. 2017, pp. 1–6.

[191] A. Heidari, M. A. J. Jamali, N. J. Navimipour, and S. Akbarpour, "Internet of Things offloading: Ongoing issues, opportunities, and future challenges," *Int. J. Commun. Syst.*, vol. 33, no. 14, p. e4474, Sep. 2020.

[192] A. B. De Souza et al., "Computation offloading for vehicular environments: A survey," *IEEE Access*, vol. 8, pp. 198214–198243, 2020.

[193] A. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 393–413, 1st Quart., 2014.

[194] S. Deshmukh and R. Shah, "Computation offloading frameworks in mobile cloud computing: A survey," in *Proc. IEEE Int. Conf. Current Trends Adv. Comput. (ICCTAC)*, Mar. 2016, pp. 1–5.

[195] N. Chalaemwongwan and W. Kurutach, "Mobile cloud computing: A survey and propose solution framework," in *Proc. 13th Int. Conf. Electr. Eng. Electron., Comput., Telecommun. Inf. Technol. (ECTI-CON)*, Jun. 2016, pp. 1–4.

[196] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Netw. Appl.*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[197] K. Peng, V. C. M. Leung, X. Xu, L. Zheng, J. Wang, and Q. Huang, "A survey on mobile edge computing: Focusing on service adoption and provision," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–16, Oct. 2018.

[198] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1584–1607, Aug. 2019.

[199] M. Rahimi, M. Songhorabadi, and M. H. Kashani, "Fog-based smart homes: A systematic review," *J. Netw. Comput. Appl.*, vol. 153, Mar. 2020, Art. no. 102531.

[200] K. Gasmi, S. Dilek, S. Tosun, and S. Ozdemir, "A survey on computation offloading and service placement in fog computing-based IoT," *J. Supercomput.*, vol. 78, no. 2, pp. 1983–2014, Jun. 2021.

[201] A. Shakarami, A. Shahidinejad, and M. Ghobaei-Arani, "A review on the computation offloading approaches in mobile edge computing: A game-theoretic perspective," *Softw., Pract. Exp.*, vol. 50, no. 9, pp. 1719–1759, 2020.

[202] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective," *Comput. Netw.*, vol. 182, Dec. 2020, Art. no. 107496.

[203] H. Wu, "Multi-objective decision-making for mobile cloud offloading: A survey," *IEEE Access*, vol. 6, pp. 3962–3976, 2018.

[204] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-UAV enabled wireless networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 2109–2121, Mar. 2018.

[205] M. Zamzam, T. El-Shabrawy, and M. Ashour, "Game theory for computation offloading and resource allocation in edge computing: A survey," in *Proc. 2nd Novel Intell. Lead. Emerg. Sci. Conf. (NILES)*, Giza, Giza, Egypt, Oct. 2020, pp. 47–53.

[206] W. Labidi, M. Sarkiss, and M. Kamoun, "Joint multi-user resource scheduling and computation offloading in small cell networks," in *Proc. IEEE 11th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Abu Dhabi, United Arab Emirates, Oct. 2015, pp. 794–801.

[207] L. Chen, J. Wu, J. Zhang, H.-N. Dai, X. Long, and M. Yao, "Dependency-aware computation offloading for mobile edge computing with edge-cloud cooperation," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2451–2468, Oct. 2022, doi: 10.1109/TCC.2020.3037306.

[208] X. Cheng, D. Duan, S. Gao, and L. Yang, "Integrated sensing and communications (ISAC) for vehicular communication networks (VCN)," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 23441–23451, Dec. 2022.

[209] Q. Liu, H. Liang, R. Luo, and Q. Liu, "Energy-efficiency computation offloading strategy in UAV aided V2X network with integrated sensing and communication," *IEEE Open J. Commun. Soc.*, vol. 3, pp. 1337–1346, doi: 10.1109/OJCOMS.2022.3195703.

Siqi Zhang (Graduate Student Member, IEEE) received the B.Sc. degree in electronic information technology from Macau University of Science and Technology, Macau, China, in 2019, and the M.Sc. degree in wireless communication from the University of Surrey, U.K., in 2020, where he is currently pursuing the Ph.D. degree in wireless communication with the 5GIC and 6GIC, Institute for Communication Systems (ICS). His main research interests include edge computing, semantic communication, goal-oriented communication, and communication and control co-design.

Na Yi (Member, IEEE) received the Ph.D. degree in wireless communications from the University of Surrey, U.K., in 2009. Her research interests include cognitive radios/networks, advanced signal processing in wireless communications and sensor networks, cooperation and relaying in wireless networks, and cognitive resource allocation for wireless networks.

Yi Ma (Senior Member, IEEE) joined the Institute for Communication Systems (ICS), University of Surrey, Guildford, U.K., in 2004, where he is currently a Chair Professor. He is the Head of the Artificial Intelligence for Wireless Communication Group, ICS, to conduct the fundamental research of wireless communication systems covering signal processing, applied information theory, and artificial intelligence. He is the Work Area Leader of the 5G and 6G Innovation Centre, University of Surrey. He has authored or coauthored more than 150 peer-reviewed IEEE journals and conference papers in the areas of deep learning, cooperative communications, cognitive radios, interference utilization, cooperative localization, radio resource allocation, multiple-input–multiple-output, estimation, synchronization, and modulation and detection techniques. He holds eight international patents in the areas of spectrum sensing and signal modulation and detection. He has served as the Tutorial Chair for EuroWireless 2013, PIMRC 2014, and CAMAD 2015. He is the Chair of the Air-Interface Club, ICS. He is the Co-Chair of the Signal Processing for Communications Symposium in ICC 2019. He was the Founder of the Crowd-Net Workshop in conjunction with ICC 2015, ICC 2016, and ICC 2017.