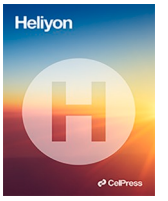
Heliyon 10 (2024) e29916

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon



Review article

A comprehensive review on internet of things task offloading in multi-access edge computing

Wang Dayong [a,*], Kamalrulnizam Bin Abu Bakar [a], Babangida Isyaku [a,b], Taiseer Abdalla Elfadil Eisa [c], Abdelzahir Abdelmaboud [d]

[a] Department of Computer Science, Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru, Johor 81310, Malaysia
[b] Department of Computer Science, Faculty of Information Communication Technology, Sule Lamido University, K/Hausa, Jigawa State, Nigeria
[c] Department of Information Systems-Girls Section, King Khalid University, Mahayil, 62529, Saudi Arabia
[d] Humanities Research Center Sultan Qaboos University, Muscat, Oman

ARTICLE INFO

Keywords:
Computation offloading
Internet of things
Mobile edge computing
Multi-access edge computing
Task offloading

ABSTRACT

With the rapid development of Internet of Things (IoT) technology, Terminal Devices (TDs) are more inclined to offload computing tasks to higher-performance computing servers, thereby solving the problems of insufficient computing capacity and rapid battery consumption of TD. The emergence of Multi-access Edge Computing (MEC) technology provides new opportunities for IoT task offloading. It allows TDs to access computing networks through multiple communication technologies and supports more mobility of terminal devices. Review studies on IoT task offloading and MEC have been extensive, but none of them focus on IoT task offloading in MEC. To fill this gap, this paper provides a comprehensive and in-depth understanding of the algorithms and mechanisms of multiple IoT task offloading in the MEC network. For each paper, the main problems solved by the mechanism, technical classification, evaluation methods, and supported parameters are extracted and analyzed. Furthermore, shortcomings of current research and future research trends are discussed. This review will help potential and new researchers quickly understand the panorama of IoT task offloading approaches in MEC and find appropriate research paths.

1. Introduction

The development of Industry 4.0 and smart cities has led to the rapid expansion of the scale of the Internet of Things (IoT) [1]. The new demands for innovation have significantly increased the computing load on terminal devices, especially IoT applications that include Artificial Intelligence (AI) functions [2,3]. However, the computing capacity and power supply of the Terminal Device (TD) are limited due to limitations of the manufacturing process, cost and battery capacity [4]. This results in limited application of IoT. The emergence of computing offloading technology provides new opportunities to solve such problems [5,6]. TDs can offload computing tasks to cloud platforms or edge computing networks, thereby significantly shortening task execution time and extending TD's battery life [7,8].

Computing networks such as cloud, fog, and edge can be used to carry offloaded tasks from TD. However, the development of 5G technology has made MECs the primary computing network to support the offloading of IoT tasks. In particular, the further

* Corresponding author.
E-mail address: wangdayong@graduate.utm.my (W. Dayong).

https://doi.org/10.1016/j.heliyon.2024.e29916
Received 20 February 2024; Received in revised form 11 April 2024; Accepted 17 April 2024
Available online 22 April 2024

development of the edge-cloud continuum architecture meets the time-sensitive requirements of task offloading. MEC technology officially released in 2016 [9,10], allows IoT devices to process data close to the data source, which reduces the latency of data transmission and helps to reduce the burden on the network [11,12]. This is particularly beneficial for IoT devices with low bandwidth or unstable network connections, it can better support IoT offloading of computing tasks.

The IoT task offloading in MEC scenario has its particularities that are different from other application scenarios. First, IoT devices need to offload more computing-intensive tasks due to very limited computing power [13]. Second, task offloading in IoT application scenarios requires optimizing the energy consumption of TD to extend the working time of the device due to the limitation of battery capacity [14,15]. Third, IoT has a larger network scale (e.g. Smart cities). This results in a huge number of TDs and edge computing nodes in the network [16,17]. Therefore, it is very difficult to find the optimal solution for task offloading decision-making. In addition, the computing tasks of IoT are highly heterogeneous due to different hardware systems and diversified software applications [18]. Moreover, IoT computing tasks are also mostly time-sensitive. Device mobility in some IoT application scenarios is also very high (such as the Internet of Vehicles, where even task migration needs to be considered). These factors jointly make the task offloading process of IoT very complex and have particularly high performance requirements [19,20]. All these lead to the fact that IoT task offloading mechanisms need to consider multiple optimization objectives comprehensively, rather than just improving a particular performance metric.

The rapid growth of the demand for multi-objective task offloading optimization coupled with advances in classical mathematical optimization, heuristic optimization and AI has expanded the research scope and capabilities in this field [15,21–23]. In particular, AI-based task offloading methods have been rapidly developed in recent years [24,25]. Such methods support more decision optimization parameters and can cope with more complex task-offloading scenarios. In addition, numerous intelligent learning models and decision-making frameworks have been used to support IoT task offloading in MEC due to the rapid development of the AI research field [26–29].

It's difficult to quickly understand the technical route and research status of the IoT task offloading mechanism in MEC due to the related research on MEC and IoT task offloading being very extensive. In addition, there is a lack of review specifically focusing on IoT task offloading in MEC. To fill this gap, this survey aims to address this critical need by providing a comprehensive overview of the state of research and development in task offloading optimization. This study comprehensively investigates IoT task offloading mechanisms in MEC network environments, analyzing the main problems addressed by the offloading mechanisms, the techniques employed, the input parameters related to the environment, the evaluation metrics of the mechanisms or algorithms, the evaluation tools, and the datasets. Our goal is to provide a holistic view of the research field.

The contributions of this study are briefly summarized below:

- This paper comprehensively reviewed the research papers on IoT task offloading mechanisms from 2016 to the present and discusses on the advantages and disadvantages of classification of implementation technologies.
- The distribution of input parameters in various offloading mechanisms was analyzed to explore the coverage and limitations of the influencing factors of the task offloading mechanism.
- Performance metrics for evaluating offloading algorithms and mechanisms are presented,
- Statistical analysis of evaluation methods, tools, and data sets for algorithms and mechanisms are presented.
- Potential research opportunities and future research directions are presented.

The rest of the paper is organized as follows: Section 2 shows the background of IoT task offloading. Section 3 describes the related works. Section 4 gives the research methodology. Section 5 presents the classification of IoT task offloading mechanisms in MEC. Section 6 discussed and answered research questions. Section 7 provides open issues and future research directions. In the end, the conclusion is provided in Section 8.

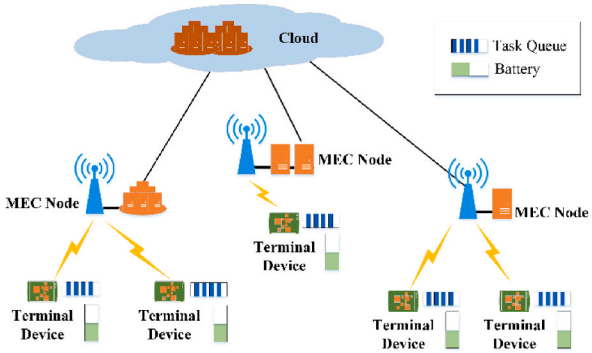


Fig. 1. Logic diagram of IoT task offloading in MEC.

2. Background of IOT task offloading in MEC

Task offloading in the context of IoT involves moving computing tasks or workloads from IoT devices to more powerful computing resources, typically at the edge or in the cloud, for more efficient processing [30,31]. Many IoT devices, especially sensors and small embedded systems, are resource-constrained with limited processing power and energy supplies [32,33]. These limitations make it challenging for them to perform complex calculations and data processing locally.

IoT applications are becoming increasingly complex and often introduce real-time analytics, image recognition, and other computationally intensive tasks [34]. These tasks be beyond the capabilities of resource-constrained IoT devices. Some IoT applications, such as self-driving cars, industrial automation, and augmented reality, require low-latency processing for real-time decision-making [35,36]. Offloading tasks to more powerful edge or cloud servers can reduce latency. However, neither the computing resources of MEC nor the communication resources of wireless networks are unlimited [37,38]. TDs still face many challenges when performing task offloading. Fig. 1 shows the logic diagram of IoT task offloading in MEC.

2.1. Edge-cloud continuum and computational task offloading

There are many computing paradigms which provide computational power for the computation offloading from capacity-constrained TDs [39]. The basic solution is to offload computing tasks directly from TD to a cloud platform with sufficient resources [40]. However, excessive network transmission delay will be introduced due to the long distance between the cloud platform and TD [41]. Instead, edge servers are deployed closer to the TD. The short communication distance greatly reduces the delay of task data transmission. However, edge servers are difficult to carry a large number of computing tasks due to insufficient computing capacity [42]. Edge-cloud continuum can solve this problem very well. In this computing paradigm, edge servers and cloud platforms jointly provide computing resources for TDs. The computing tasks offloaded from TDs are first performed by the edge server, and part of the offloaded tasks will be transferred to the cloud platform for execution when the edge server power is insufficient [43]. Thus, the performance of task offloading can be significantly improved due to taking full advantage of the fast response from edge servers and the sufficient resources provided by cloud platform. Fig. 1 presents the architecture of IoT task offloading in edge-cloud continuum.

In large-scale networks, tasks can be offloaded vertically to edge servers and clouds, or horizontally between different edge servers [44]. In addition, the methods for offloading computing tasks between TDs have also been explored [45,46]. The differentiation of computing task characteristics and the dynamic changes in MEC available resources lead to increased volatility and complexity of the computing offloading process [47]. In addition, the computing task can be either treated as a complete unit for binary offloading or can be split into small subtasks for partial offloading [48,49]. Therefore, it is necessary to optimize the offloading process with the help of offloading decisions and resource allocation, thereby reducing latency and energy consumption.

2.2. Task offloading decision

The main goal of TDs to offload computing tasks is to improve task execution efficiency and reduce their energy consumption [39, 40]. However, transmitting tasks in congested wireless networks will introduce higher communication delays. In addition, tasks to perform IoT offloading on busy MEC nodes need to be queued for a long time. If it takes longer to execute a task through offloading, TD will not be able to improve the execution efficiency of the task during the task offloading process. Similarly, wireless data communication will consume part of the TD's energy [41,42]. If the communication energy consumed by the transmission task is greater than the energy consumed by the local execution of the task, TD should not perform task offloading. Therefore, before undertaking task offloading, it is necessary to evaluate whether there will be benefits from offloading tasks in the current state.

2.3. Task assignment

Task offloading decisions are often made dynamically based on factors such as device capabilities, network conditions, and application requirements [43]. The IoT ecosystem is highly heterogeneous and contains a variety of devices with varying levels of computing power. The task offloading mechanism needs to jointly analyze the available resource status of the MEC network, the current working status of the communication network, and the constraints of the tasks to be offloaded to make decisions to offload IoT tasks to appropriate remote computing nodes. In some simple application scenarios, computing tasks only need to be offloaded to servers in the MEC network [44,45]. In some complex application scenarios, computing tasks be offloaded to MEC servers or cloud platforms [46], or even to other idle TDs [47]. This makes the choice of destination for task offloading extremely complex. If we consider the attribute differences of tasks and the requirements for high-density parallel task offloading, the complexity of task offloading allocation will further increase.

2.4. Resource allocation

Considering that in many application scenarios, task offloading requests are often greater than the services that the MEC network can provide [48–50], limited MEC resources need to be appropriately allocated to numerous offloading tasks based on differences in task requirements. The resources that need to be allocated mainly include the computing resources of the MEC and the available communication resources in the network [51–53]. Reasonable resource allocation can maximize the overall efficiency of task offloading and minimize the proportion of tasks over time. In addition, due to the limitation of TD's power supply capacity, the impact of

the allocation strategy on TD's energy consumption also needs to be considered when allocating resources [54]. These factors will increase the complexity of resource allocation, and the real-time requirements for IoT task offloading are relatively high. Therefore, allocating resources efficiently and in real time is one of the issues that need to be solved urgently.

2.5. Global optimization and multi-objective optimation

The global optimization problem of IoT task offloading in a large and heterogeneous MEC network is complex because it involves multiple goals and the effective allocation of global resources, and one cannot just focus on the task offloading effect of a certain TD. Multiple MEC servers and IoT devices be distributed globally [55,56]. The global optimization problem involves how to coordinate task offloading and resource allocation among these distributed resources, thereby maximizing the completion speed of task offloading for multiple TDs and the overall utilization of MEC resources [57,58]. Global optimization problems also need to consider collaborative decision-making among multiple decision-makers, including IoT devices, MEC servers, network operators, etc. Distributed offloading decision-making architecture is often used in some large-scale application scenarios [59,60], which requires full consideration of reasonable global decision-making optimization.

Multi-objective problems of IoT task offloading often involve trade-offs between minimizing latency and minimizing energy consumption [61]. Latency is a key performance metric, and energy management is critical to the longevity and efficiency of IoT devices. In addition, in multi-objective problems, it is necessary to ensure that the Quality of service (QoS) of IoT tasks is satisfied [62,63]. This include metrics such as minimizing packet loss or maximizing bandwidth utilization [64,65]. Besides this, another goal be the efficient distribution of tasks among multiple MEC servers to ensure that resources are fully utilized.

Solving these problems often requires advanced optimization techniques. At the same time, factors such as the actual deployment environment, application requirements, and network topology also need to be comprehensively considered. Studying and optimizing solutions to these problems can help achieve efficiency, performance, and reliability of IoT task offloading while meeting multiple optimization goals. Fig. 1 shows the logic diagram of IoT task offloading in the MEC network environment.

3. Related works

The field of computation offloading has seen significant growth in recent years, with numerous studies and advancements in various sub-domains. In this section, we provide an overview of relevant research and reviews that have contributed to the current state of the field.

A review of computation offloading that comprehensively investigates intelligent computational task offloading methods in MEC [66]. The study formulated the costs of local computing and remote computing respectively, and then proposed a unified computation offloading process model. In addition, task offloading optimization methods based on multiple AI techniques were compared in multiple dimensions, and a more complete study of metrics for MEC computational offloading was conducted. Further, the study also points to efficient service discovery, flexible computational task splitting, and security as future research directions. However, the articles investigated in this work are not IoT-specific, but broadly discuss intelligent optimization techniques for mobile applications to offload computational tasks to MEC networks. In addition, the survey only discussed AI-based computing task offloading optimization methods and did not involve methods such as heuristics and classical mathematical optimization. These methods are still evolving.

While the work in Ref. [67] the task offloading paradigm in the MEC environment is systematically reviewed from a technology evolution perspective. Especially, MEC is distinguished from ordinary edge computing based on MEC's unique support for device mobility. In addition, quantitative analysis methods and qualitative analysis methods were combined in this study, and the problem model, solutions, and evaluation indicators of each article were extracted and summarized. Thus, future research directions are given. However, this study discussed a lot about the MEC architecture and features but did not focus on task offloading optimization. In addition, most of the articles surveyed were not in the IoT context.

The computing task offloading decision-making is explored in Ref. [68]. This review paper investigated the research results related to IoT application models and focused on the optimization of task division during the computing offloading process. This study treats various types of applications differently and discusses various task-splitting optimization methods in the computing offloading process. In addition, this paper also proposes a comprehensive classification metric for evaluating computation offloading approaches that support task segmentation. Furthermore, this study also provides future research directions for IoT in the cloud-edge computing paradigm. However, the research lacks focus on the MEC architecture but rather on cloud-edge. Thus, the characteristics of MEC such as mobility are ignored.

The research progress and ongoing issues of IoT computing task offloading mechanisms are systematically reviewed in Ref. [69], and a feature comparison between various offloading mechanisms is provided. In this paper, the task-offloading mechanism is divided into two categories: static and dynamic based on different task-offloading decision-making methods. In addition, this study discusses the advantages and disadvantages of two different types of offloading mechanisms in various application scenarios. In addition, a new classification method based on offloading decision-making mechanisms and overall architecture is proposed. However, there are more discussions about offloading tasks to the cloud, fog computing networks and traditional edge computing networks, but the articles are not in the MEC context and lack attention to terminal mobility.

A whole bunch of enabling technologies for IoT task offloading in fog-cloud computing environments are studied in Ref. [70]. In addition, a variety of technologies that support task offloading are discussed according to differences in application scenarios. This study reviews computing task offloading methods based on cloudlet, mobile edge computing, micro-datacenter, and nano-datacenter respectively. Moreover, the Scalability and Service level agreement (SLA) of task offloading networks are considered challenges for

future research. However, the article rarely discusses the particularities of MEC task offloading, nor does it consider the characteristics of other computing platforms. Instead, it discusses more about IoT task offloading based on the concept of macro fog computing.

In [71] the computation offloading methods in Fog-based IoT are reviewed, and the computing task offloading network is divided into two levels: IoT to Fog and Fog to cloud. In this paper, task offloading mechanisms are discussed separately at different computing network levels. Similarly, this study also distinguishes different data scales of IoT computing offloading and studies the adaptability and deficiencies of various computing offloading mechanisms in application scenarios of different computing scales. Finally, open issues and future research challenges based on the three-layer task offloading mechanism are discussed. However, selected articles are not focused on MEC, but on fog context. In addition, this research focuses on Machine Learning (ML) based computing task offloading optimization methods but lacks discussion of other types of task offloading optimization methods.

A survey of MEC-enabled mobile user computing offloading technology proposed a new classification of MEC internal task offloading strategies [72]. In this study, researchers not only discussed the strategy for user equipment to offload computing tasks to MEC but also focused on the optimization of task offloading between computing resource nodes in the MEC network. Also, offload optimization methods are divided into user device-driven, cloud-assisted, SDN-based, and Collaborative types. Consequently, the characteristics and shortcomings of various classified offloading optimization methods are compared and analyzed respectively. In the end, technical challenges and future research directions are briefly discussed. However, selected articles are not in the IoT context, but more about the optimization methods of common computing task offloading in MEC.

Some task offloading solutions in mobile edge computing networks are reviewed in Ref. [24], and a model of the process of executing computation offloading for users is given. According to different optimization methods for task offloading in MEC, this study compared the characteristic and application scenario adaptability of traditional mathematical solving methods, heuristic methods, and reinforcement learning methods. The main task offloading optimization objectives considered in this study include latency, energy consumption, comprehensive offloading benefits, and system resource utilization. Meanwhile, this study also discusses the impact of partial task offloading, resource allocation, and inter-task dependencies on task offloading decisions. Finally, real-time environment awareness and security issues are considered challenges for future research. However, most of the articles do not engage with IoT and there is no discussion related to performance evaluation of task offloading.

According to the above discussion, we summarize and find that there are two main weaknesses in existing research as shown in Table 1. One of the main weaknesses is that the articles selected in the review are not related to IoT. Many studies have been conducted on the optimization of task offloading by mobile user devices in MEC environments. However, numerous studies for IoT task offloading

Table 1
Summary of related works.

Ref.	Main work	Advantage	Weakness
[66]	Investigated the computing offloading methods and key indicators in MEC, and focused on the optimization of computing offloading based on artificial intelligence technology.	A comprehensive survey of computational offloading methods based on AI techniques.	Articles are not relevant to the IoT. The process of literature selection is not transparent.
[67]	The Task offloading paradigm in MEC was systematically reviewed, and three main task offloading optimization routes were identified based on quantitative analysis methods.	Distinguishes between MEC and regular edge computing in terms of computational offloads.	Most of the articles are not relevant to the IoT.
[68]	The research results related to IoT application models and computing task offloading decision engines were investigated, with a focus on task division optimization during the computing offloading process.	Comprehensive classification metrics are proposed for joint optimization of different application types.	Lack of focus on the mobility of MEC but rather on cloud-edge. The process of literature selection is not transparent.
[69]	This paper systematically reviews the research progress and ongoing issues on the mechanism of IoT computing task offloading and proposes future research challenges. A parameter comparison between offloading methods is also provided.	A new classification method based on offloading decision-making mechanisms and architectures is proposed.	Articles are not relevant to the IoT.
[70]	Researched enabling technologies for IoT task offloading in a fog computing environment. Some standards for task offloading between fog and cloud layers are proposed.	The differences in task offloading in different network layers are distinguished.	Articles are not focused on MEC. The process of literature selection is not transparent.
[71]	Reviewed machine learning technology for computation offloading in Fog-based IoT, and discussed the challenges and opportunities of computing offloading in IoT to Fog and Fog to cloud environment.	Different data volume scales for IoT computing offloading are distinguished.	Articles are not focused on MEC. The process of literature selection is not transparent.
[72]	The computing offloading strategy of user equipment in MEC is discussed, and the task offloading between computing resource nodes in MEC is focused on.	A new classification of MEC internal task offloading strategies is proposed.	Articles are not relevant to the IoT. The process of literature selection is not transparent.
[24]	Reviewed various methods of task offloading in MEC, traditional mathematical solution methods, heuristic methods, and reinforcement learning methods, and discussed partial offloading, resource allocation, and task dependency.	A model of the Process of executing computation offloading for users is given, and the impact of task dependencies on task offloading decisions is discussed.	Most of the articles are not relevant to the IoT. The process of literature selection is not transparent.

have not considered the characteristics of MEC but rather discussed it together with ordinary edge computing and fog computing. Thus, there is no comprehensive review focused on IoT computing task offloading in MEC environments. The goal of this study is to systematically review the research progress, ongoing issues, and future directions for further research in IoT task offloading optimization in MEC network environments.

Briefly, there are some weaknesses in the work mentioned above as follows:

- Some papers do not provide IoT task offloading in MEC
- Some papers do not include MEC features for IoT task offloading
- Some papers do not distinguish the differences between MEC and traditional Edge computing
- Some papers do not differentiate between IoT task offloading and ordinary computation offloading
- Some papers do not provide IoT task offloading in the MEC selection process for the article

The above-mentioned explanations were the inspiration and motivation to organize a survey paper on IoT task offloading in MEC to overcome all of these lacks.

4. Research methodology

The study formulated some research questions which were used to search various data sources using different keywords. The search process was conducted in various databases based on inclusion and exclusion criteria. Afterward, the study assesses the quality of the papers through the article selection and screening process.

The research methodology employed in this review is designed to ensure a rigorous, transparent, and replicable process for identifying, screening, and synthesizing relevant literature. The methodology follows established guidelines and best practices for conducting literature reviews, which aim to minimize bias and enhance the validity of the findings.

A deep search is conducted by seeking keywords related to IoT task offloading in MEC. The selection time range of the literature is set from 2016 to the current time because the MEC specification was officially released in 2016 [9]. Next, a multi-level screening of the collected literature was conducted based on titles, keywords, and abstracts. In the next step, the screened literatures were scored according to the set rules, and 138 eligible literatures were finally selected for in-depth analysis. Fig. 2 presents the year-wise publication of IoT task offload in MEC, which is selected for the study.

4.1. Research questions

In order to have a comprehensive understanding, this study investigates several aspects such as task offloading mechanism, network environment support, task offloading architecture, evaluation tools, and data. The primary research questions for this paper are listed below:

- RQ1: What main problems does the IoT task offloading mechanism solve?
- RQ2: What are the main goals of the offloading mechanism?
- RQ3: What technologies are based on various mechanisms and algorithms to achieve offloading optimization?
- RQ4: What network access methods does the offload mechanism support in MEC?
- RQ5: Which MEC network architectures can the offloading mechanism work on?
- RQ6: How many target compute nodes are tasks offloaded?
- RQ7: What are the main input parameters considered by the offloading mechanism?
- RQ8: What methods are used to implement the evaluation of offloading mechanisms?
- RQ9: What baselines are mainly compared in the evaluation of offloading mechanisms?
- RQ10: What metrics are used to evaluate the performance of the offloading mechanism?
- RQ11: Which datasets are used to evaluate offloading mechanisms?

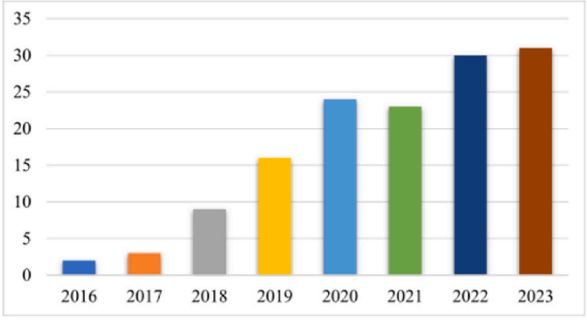


Fig. 2. Articles selected for review are published year wise.

4.2. Keywords

To identify relevant literature, a search strategy is meticulously constructed. This strategy includes a set of carefully chosen keywords, Boolean operators, and search strings tailored to the research topic. The goal is to cast a wide net across selected databases while maintaining specificity and precision in the search. These keywords used in searching are presented in Table 2.

During the keyword-based literature retrieval process, the selected keywords are divided into three categories: IoT, edge computing environment, and optimization of task offloading. For each category, multiple different keywords with similar meanings are used, and then the Boolean-based document retrieval of command is created. After that, multiple retrieval restrictions such as time range and language are set. The following are the advanced search commands with the literature retrieval of the IEEE Explore as an example.

((“All Metadata”:Internet of things) OR (“All Metadata”:IoT)) AND ((“All Metadata”:Edge computing) OR (“All Metadata”:Mobile edge computing) OR (“All Metadata”:Multi-access edge)) AND ((“All Metadata”:Computation offloading) OR (“All Metadata”:Task Offloading)) AND ((“All Metadata”:Offloading Decision) OR (“All Metadata”:Offloading optimization) OR (“All Metadata”:IoT Task scheduling) OR (“All Metadata”:MEC-IOT))

4.3. Inclusion and exclusion criteria

Inclusion and exclusion to filter the most suitable research literature on IoT task offloading in MEC are given in Table 3.

4.3.1. Inclusion and Exclusion Criteria

From the articles which are selected for review, Fig. 4 shows the number of articles published year-wise, which are selected for the study.

The articles are categorized based on the methods they have used as Classic mathematical, Lyapunov optimization, Heuristic, Game theory, and AI-base IoT task offloading techniques in MEC. Out of these 83 articles were from IEEE, 18 articles from Science Direct, 7 from Springer, 6 from ACM, 5 from Elsevier, 3 from MDPI, 1 from Willey, and 15 from others.

4.4. Quality assessment

The methodology employed in this review is underpinned by a commitment to minimizing bias, ensuring the inclusion of relevant literature, and providing a strong foundation for the analysis. It is designed to yield a comprehensive and reliable synthesis of the existing research in the field. The quality and relevance of included articles are assessed to ensure the integrity of the review. Quality assessment tools, where applicable are used to evaluate the methodological rigor of primary studies. Table 4 shows the strategy for scoring article quality.

4.5. Article selection and screening

A multiple-step screening process is implemented. Initially, articles are screened based on titles and abstracts against predefined inclusion/exclusion criteria. Subsequently, full-text articles that pass the initial screening are assessed against the same criteria to determine their eligibility for inclusion.

The researchers minimized the possible impact of potential information bias by ensuring that the exercise was a rigorous review of the literature, carefully selecting the sources and selection criteria for the data and using software tools to verify the data entries thereby avoiding duplication of data.

Four researchers (authors of this paper) were involved in the strategies developed to minimize selection bias. The process of literature screening was conducted independently by four researchers in a stepwise manner. The first researcher conducted the first search using a predefined search equation and subsequently performed the process of eliminating duplicate documents. The second researcher re-examined the results obtained in the first search and performed a second exclusion based on the criterion of document relevance to the topic. The third researcher analyzed the body of the search results in depth and publicly answered the previously defined research questions with her researcher. The fourth researcher organized the review of the obtained results by all participants of

Table 2
Keywords used in searching.

S#	Keyword	Broader term
1	Internet of things/IoT	IoT computing architecture
2	Edge computing	Edge computing environment
3	Mobile edge computing	MEC features
4	Multi-access edge computing	MEC features
5	Computation offloading	Computation offloading in MEC
6	Task Offloading	Task offloading algorithm and mechanism
7	Offloading Decision	Decision-making methods for offloading
8	Offloading optimization	The multi-objective offloading optimization mechanism
9	IoT Task scheduling	Task scheduling methods
10	MEC-IOT	IoT features in MEC

Table 3
Inclusion and exclusion criteria.

Inclusion Criteria	Exclusion Criteria
The study focuses on IoT task offloading	The study that focuses on other computation offloading issues
The study focuses on task offloading in MEC network	Task offloading in cloud, fog computing, and cloud-let are not considered
Published papers since 2016	Articles before ETSI officially released the MEC specification in 2016 are not considered
Publications written in English	The articles written in other languages are not considered
Peer-reviewed publications in high-quality journals and conferences that can be accessed	Disreputable and inaccessible articles are not considered

Table 4
Articles quality score.

Factor	Points	Description
Topic	0–4	Strong correlation
Algorithm or mechanism	0–4	A complete and clear description
Performance metrics	0–4	Clear definition
Experiment	0–4	A complete and clear description
Benchmark	0–4	Explicit comparative analysis
WOS ranking	0–4	Q1 = 1, Q2 = 2, Q3 = 3, Q4 = 4, none = 0

this study.

5. IOT task offloading mechanism in MEC

Based on detailed investigation, the IoT task offloading mechanisms in MEC are classified based on technical principles. The wide variety of offloading mechanisms are divided into the Classic mathematical optimization approach, Lyapunov optimization based approach, Heuristic based approach, Game theory based approach, and AI-based approach. In addition, multiple technical methods have been involved in combination in some studies. Fig. 3 shows the taxonomy of IoT task offloading mechanisms in MEC based on different technologies.

5.1. Classic mathematical optimization based offloading mechanisms

The task offloading mechanism based on classical mathematical optimization is the most basic IoT task offloading coordination method, aiming to optimize the distribution of computing tasks in heterogeneous computing environments. These mechanisms rely on mathematical optimization techniques to determine which tasks should be offloaded to remote resources, taking into account factors such as network conditions, device capabilities, and user preferences. By formulating the offloading decision as a mathematical optimization problem, these methods aim to find the most efficient task allocation, providing rigorous and optimal solutions to task allocation challenges.

There are a great number of articles that study classic mathematical optimization based IoT task offloading mechanisms. Some of

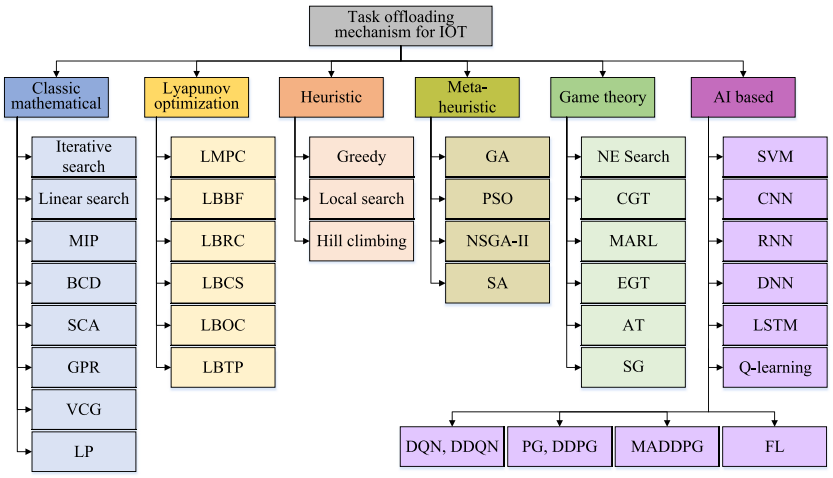


Fig. 3. Taxonomy of IoT task offloading mechanism in MEC.

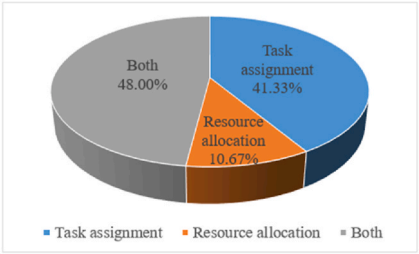


Fig. 4. Distribution of problems addressed of IoT task offloading in MEC.

Table 5
A side-by-side comparison of offloading mechanisms based on classic mathematical optimization.

Ref.	Problems addressed	Technique	Network	Objective	Evaluation tools	Baseline	Advantages	Limitations
[73]	Task assignment	BCD	OFDMA, WPT	Maximize total computation rate	Python based implementation	ES, BA, EC, LC	Joint optimization of transmit power of the GW, backscatter coefficient, TS ratio, and computing mode	The impact of TD mobility on energy consumption optimization is not considered
[74]	Task assignment	Lagrangian multipliers, mixed-integer programming	Mixed	Minimize total network delay and network price	Testbed	SDTO, FLAVOUR	Provide fair QoS	Inadequate consideration of wireless network characteristics
[75]	Task assignment, resource allocation	Hierarchical iterative search	TDMA	Minimize energy consumption	Matlab	JCCO, CPLEX, HD,	Tradeoff between transmission efficiency and MEC utilization	Collaborative offloading of multiple MEC nodes is not supported
[76]	Task assignment, resource allocation	SCA	RAN	Minimize latency and energy consumption	Matlab	Random, UAV-Only, EC-Only, Fixed UAV-EC	Jointly optimizing UAV position, communication and computing resource allocation, and task-splitting decisions	Multi-UAV collaborative uninstallation is not supported
[77]	Task assignment	GRP	RAN	Minimize latency and energy consumption	Matlab	MOEA/D-DE, MOEA/D-DRA, MOEA/D-CMA, MOEA/D-PaS	Higher exploration ability and better generalization	Concurrent task offloading is not considered
[78]	Resource allocation	Iterative search	NOMA	Maximize energy efficiency	Testbed	OMA, NOMA-EPA	Joint radio and computation resource allocation	Task offloading between TDs is not supported
[79]	Task assignment	SCA	RAN	Minimize latency and energy consumption	Matlab	DMin-SCO, EMin-SCO	Taking each user's satisfaction into account	Collaborative offloading of multiple MEC nodes is not supported
[80]	Task assignment, resource allocation	Linear search	RAN	Minimize overall delay	Matlab	Solver of integer programming from MATLAB	Joint optimization of the secrecy-provisioning, computation offloading, and radio resource allocation	Collaborative offloading of multiple MEC nodes is not supported
[81]	Task assignment	Linear search	RAN	Minimize overhead	Matlab	Unsecure LBCO, Secure LBCO, CO, LE, FO	Balance the loads among BSs	TD movement between BSs is not supported
[82]	Resource allocation	Lagrangian multipliers, LP, critical-value, VCG	RAN	ILOG CPLEX	Matlab	BDA, DPDA, McAfee, OPTIMAL	Acceptable simple framework	Does not support large-scale network scenarios and does not consider the impact of the network environment

them are presented in Table 5. In Ref. [73] the problem is decomposed into two sub-problems: time split ratio optimization and offloading decision optimization. The authors proposed an iterative algorithm based on the block coordinate descent (BCD) method. The algorithm consists of two main steps: exhaustive search (ES) and the proposed fast exhaustive algorithm (FEA). The generation of an offloading decision matrix and the allocation of communication resources are realized respectively. A mathematical framework based on Lagrangian multipliers is proposed in Ref. [74]. The framework is designed for dynamically scheduling microservices to optimize network latency and price costs of task offloading. The goal is to maximize energy efficiency, provide fair quality of service (QoS), minimize network latency and price, and improve satisfaction levels, energy consumption rates, failure rates, and network throughput.

Research focusing on the energy consumption of networked nodes is conducted in Ref. [75]. They study the joint optimization problem of task offloading, transmission power, time, and computing resources of full-duplex communication in MEC-aided cellular IoT. The coupling constraint problem is solved through a mixed integer nonlinear programming method. For the decoupled case, the optimal conditions for transmission power and computation offloading are derived, and the optimal time allocation is obtained using the Karush-Kuhn-Tucker (KKT) condition. For the coupled case, the problem is reformulated as a two-level knapsack problem, and an efficient algorithm is developed to solve it. This solution is then used to modify the solution for the decoupled case to ensure that the coupling constraints are met.

In [76] the IoT task offloading process is modeled as a non-convex optimization problem. The goal is to minimize the service delay weighting and UAV energy consumption of all IoT devices by jointly optimizing task splitting decisions, UAV locations, communication bandwidth allocation, and computing resource allocation of UAVs and ECs. To solve this challenging problem, the authors propose an algorithm based on continuous convex approximation.

A UAV hybrid MEC system is studied in Ref. [77] that realizes the cooperation of edge cloud and UAV to achieve flexible computing offloading. The computation offloading problem is modeled as a multi-objective optimization process taking into account latency, energy consumption, and server cost. The authors propose an intelligent computational offloading algorithm based on a comprehensive optimization framework, which includes a mixed integer transformation solving framework, an improved multi-adaptive MOEA/D-DE algorithm, and a gray relational projection (GRP) method for selecting the optimal compromise Offload decision.

In the application scenario of large-scale IoT device access, the characteristics of the wireless access network can be used to optimize task offloading [78]. The goal of this study is to propose a NOMA-based energy-efficient MEC design for multi-cell networks with IoT devices and formulate a joint communication and computing resource allocation problem to maximize energy efficiency and ensure timely task execution. This scheme uses the nearest base station association strategy to associate each IoT device with the nearest SBS that provides the maximum average channel gain. It then continuously and iteratively optimizes sub-channel allocation, transmission power allocation, and computing resource allocation to maximize energy efficiency.

A task offloading method that maximizes user demand satisfaction has been proposed in Ref. [79], which considers offloading rate, tolerable delay, task workload, and maximum power constraints by formulating an optimization problem. The proposed scheme obtains suboptimal solutions to non-convex optimization problems based on the algorithm of continuous convex approximation (SCA). In Ref. [80], the optimization goal is to minimize the total delay in completing a computational task taking into account confidentiality requirements and energy consumption constraints. The proposed method transforms the overall optimization problem into multiple sub-problems and solves them iteratively. In Ref. [81] the research aims to propose a comprehensive load balancing and computing offloading technology to solve resource constraints, latency, and security issues in multi-layer IoT and edge cloud computing systems. The problem is constructed as a single optimization task with a linear objective function and constraints, and the branch-and-bound method is used to find the optimal solution. However, this method can only decide whether the task should be executed locally or offloaded to a remote node for execution.

In [82], a true combinatorial two-way auction mechanism called TCDA for the resource exchange process in the Industrial Internet of Things (IIoT) is presented. This mechanism takes into account the locality constraints of the MEC system. The mechanism uses a linear programming-based filling method for allocation, critical value-based pricing strategy, and VCG-based pricing strategy to achieve authenticity and budget balance of MDs and ESs. The ultimate goal is to maximize social welfare under locality constraints.

Such methods are based on iterative search mechanisms and are therefore not suitable for large-scale networks. Otherwise, a large number of iterations will be required to find the optimal solution, and the computational cost will increase significantly. In addition, such methods cannot comprehensively analyze too many input parameters, which makes it difficult for such methods to describe complex mobile edge computing systems and can only achieve relatively simple task offloading optimization.

5.2. *Lyapunov optimization based offloading mechanisms*

The task offloading mechanism based on Lyapunov optimization represents a type of offloading strategy commonly used in wireless communication networks and resource-constrained environments. These mechanisms leverage Lyapunov optimization to make real-time decisions about task offloading and resource allocation. Lyapunov-based task offloading mechanisms are good at managing the inherent trade-offs between conflicting goals. They cleverly allocate resources to tasks, weighing the need to reduce latency against the need to conserve energy. By continuously optimizing Lyapunov functions, they can adapt in real-time to achieve the proper balance.

A dynamic computing resource allocation algorithm based on Lyapunov optimization is proposed in Ref. [83]. This algorithm independently optimizes the offloading decision of TD and the calculation management of the MEC server. Minimize the average timeout probability by managing resources and selectively discarding tasks. The algorithm does not require a priori knowledge of task running time costs.

In [84], the research problem is transformed into a deterministic optimization problem. The author proposes a virtual queue model

and a Lyapunov online energy consumption optimization algorithm to balance the backlog and energy consumption of the task offloading queue. Moreover, the overhead of wireless transmission is considered in the offloaded transmission model.

While the work in Ref. [85] an offloading decision-making model for the IoT edge cloud computing model based on a weighted call graph is presented, which determines the computing location of tasks based on the computing cost. This mechanism can allocate tasks to MEC servers and cloud platforms at the same time. This mechanism uses an offloading decision matrix to represent offloading decisions, which results in an exponential increase in the number of possible combinations.

A parallel offline processing method based on Lyapunov optimization is presented in Ref. [86], which achieves joint optimization through cubic decoupling objective functions. This study focuses on considering the delay sensitivity of computing tasks and the energy consumption saving of the system. In addition, a MEC heterogeneous network system model is constructed in Ref. [87] that supports 5G communication characteristics and proposed a dynamic task offloading optimization scheme based on a combination of queuing theory and Lyapunov optimization. In addition, the study considers static and dynamic sub-channels and uses Lyapunov optimization and simulated annealing genetic algorithm (SAGA) for static sub-channels, and SAGA and sequential quadratic programming (SQP) for dynamic sub-channels.

In [88], the perturbed Lyapunov optimization method is adopted. The perturbed Lyapunov function and drift plus penalty function are defined, and a knapsack problem is solved for each time slot to obtain optimal scheduling. This method dynamically analyzes the available computing resources of TD and MEC servers to minimize the length of the task offloading queue and maximize system utility.

A global energy consumption optimization of task offloading based on Lyapunov optimization theory is implemented in Ref. [89]. In addition, this study also focuses on multi-task parallel offloading for each TD, rather than just offloading multiple computing tasks in a serial sequence manner.

The principle of this type of task offloading optimization method is to use Lyapunov functions to model and analyze the system state, and to achieve a dynamic allocation of resources by constructing Lyapunov drift to meet performance indicators and constraints. The Lyapunov optimization method brings significant advantages in IoT task offloading. It can ensure the stability of the system. By constructing the Lyapunov function and Lyapunov drift, the system can adaptively maintain a stable state, which is crucial for application scenarios that require high system stability. In addition, the Lyapunov optimization method can effectively guarantee performance, such as latency and throughput, and meet performance requirements while fully considering system resource constraints. This makes it suitable for various IoT application scenarios and improves the adaptability of the system. However, the complexity and computational overhead of the Lyapunov optimization method are its main challenges, requiring in-depth mathematical and engineering knowledge, as well as balancing performance guarantees with computational resources. Table 6 presents a side-by-side comparison of offloading mechanisms based on Lyapunov optimization.

Table 6
A side-by-side comparison of offloading mechanisms based on classic Lyapunov optimization.

Ref.	Problems addressed	Technique	Network	Objective	Evaluation tools	Baseline	Advantages	Limitations
[83]	Task assignment, resource allocation	Lyapunov optimization	LTE	Minimize average timeout probability	Implementation with TensorFlow	Standard queuing	Acceptable framework	Parallel uninstallation is not supported
[84]	Task assignment	Lyapunov optimization	RAN	Minimize energy consumption	Matlab	Local-only, edge-only	High stability	Not included in the task deadline
[85]	Task assignment	Lyapunov optimization	RAN	Minimize energy consumption and delay	Implementation	IoT-Only, Edge-Only, Cloud-Only, LARAC, EEDTO	Automatically distinguish task types	Ignores the differences in characteristics between different types of wireless networks
[86]	Task assignment	Lyapunov optimization	5G	Minimize system cost and task drop ratio	Matlab	LODCO, OEA, DOA	Ensure the robustness of task processing	Resource utilization is not considered
[87]	Task assignment, resource allocation	Lyapunov optimization	5G	Minimize energy consumption and latency	Matlab	Genetic algorithm, simulated annealing algorithm	Make full use of 5G communication features	
[88]	Task assignment	Lyapunov optimization	RAN	Maximize network utility balancing throughput and fairness	Matlab	Round Robin, Proportional Fair	Self-learning network fluctuations	Collaborative offloading of multiple MEC nodes is not supported
[89]	Task assignment, resource allocation	Lyapunov optimization	RAN	Minimize response time and dropping tasks	Matlab	Greedy, 0–1 offloading, complete edge, complete local	Jointly optimized for fast response and energy sustainability	Collaborative offloading of multiple MEC nodes is not supported

5.3. Heuristic based offloading mechanisms

Heuristic-based task offloading mechanism is a type of offloading strategy aim to provide an optimal solution in a short time. It is a type of algorithm based on intuition or experience, which provides a feasible solution for each instance of the combinatorial optimization problem to be solved at an acceptable cost (referring to computing time and space), and the degree of deviation of the feasible solution from the optimal solution.

The work [56] presents a heuristic algorithm that is aimed at finding a near-optimal scheduling solution for a given number of task offloading requests from TDs to a given set of MEC servers. The algorithm first calculates the number of offloading requests accepted based on the data size of the computing task. The probability is based on existing historical records. Second, try to distribute computing tasks to multiple edge servers with the goal of minimizing latency. In addition, the algorithm continuously improves the strategy based on benchmark feedback of execution results during the iterative process of task allocation.

A simple offloading mechanism is developed in Ref. [90] that jointly considered the deadline requirement of the task and the energy consumed by the device, and modeled the task offloading problem as MINLP. The response time and energy consumption are minimized by scheduling the distribution of offloading tasks in the MEC network. However, this mechanism does not consider the mobility of TD.

An enhanced version of the opportunity cost-based offloading algorithm is proposed in Ref. [91]. This algorithm supports offloading computing tasks to MEC and cloud platforms and makes full use of the characteristics of the 5G communication environment to optimize task offloading.

An integer linear programming (ILP) and approximation algorithm for problems without bandwidth capacity constraints is presented in Ref. [92]. Afterward, a greedy algorithm is proposed to solve the bandwidth capacity constraint problem. The algorithm decides whether to accept the task offloading request initiated by IoT based on the computing resource and bandwidth resource cost model.

The task offloading problem in the IoV is formalized as a set of coverage problems in Ref. [93]. Additionally, a submodule optimization framework is proposed to derive an optimal set of collected images to minimize data redundancy and maximize coverage of the reconstructed road scene. This method jointly considers cost and delay constraints to allocate the tasks to be offloaded to nearby MEC computing nodes.

An SDN-based task offloading architecture for industrial IoT is proposed in Ref. [94], and the optimization problem is divided into multiple sub-areas or communities. The distributed industrial IoT controller and the edge orchestration module coordinate and process task offloading requests based on the network available resource information provided by SDN, thereby meeting the strict latency requirements of industrial IoT tasks.

In [95], the proposed Min-CCV and Min-V algorithms search for computing nodes that meet the requirements based on minimizing delay and default cost until a suitable target server is found. However, this method is difficult to achieve good task allocation results in scenarios with insufficient computing resources.

Table 7
A side-by-side comparison of offloading mechanisms based on heuristic.

Ref.	Problems addressed	Technique	Network	Objective	Evaluation tools	Baseline	Advantages	Limitations
[56]	Task assignment	Probability distributions	Undefined	Minimize computational time	Python based implementation	CPLEX	Global knowledge utilization	Ignore energy consumption
[90]	Task assignment	Greedy	Undefined	Maximize deadline satisfaction ratio	Python based implementation	NECS, Detour	Joint optimization of deadlines and energy consumption	Not suitable for large networks
[91]	Task assignment, resource allocation	Greedy	5G	Maximize utility, Minimize energy consumption	Testbed	UGA, DFA	Tradeoff between utility and energy consumption	Parallel task offloading of a single TD is not supported
[92]	Task assignment	Greedy	Wifi	Maximize total utility	GT-ITM	ILP, classic greedy algorithm	Support cloudlets	Not suitable for large-scale networks
[93]	Task assignment	Greedy	X2 link	Maximize cost while the latency of tasks is guaranteed	NS-3	MC-RDA, MC-UNA, MC-GA	Reduce the redundant data	Task offloading between TDs is not supported
[94]	Task assignment, resource allocation	Greedy	SDN	Minimize response latency	Python based implementation	Local_NO_SDN, GATO, Max_C, Min_Q	Features that support SDN	Not suitable for large-scale networks
[95]	Task assignment	Linear search	RAN	Minimize latency and villation cost	Matlab	RR, TcaS, Random	Lightweight algorithm	Ignore server overload situation

Task offloading optimization methods based on heuristic algorithms can obtain approximately optimal decisions at low computational costs. However, such algorithms need to work in conjunction with specific problem characteristics. Therefore, the scalability and flexibility of such methods are limited. In addition, task offloading decision-making methods based on heuristic algorithms are usually difficult to support large-scale MEC network environments due to the slow convergence speed. Table 7 presents a more detailed comparison of task offloading and resource allocation schemes based on heuristic algorithms.

5.4. Meta-heuristic based offloading mechanisms

Meta-heuristic algorithm is a generalized form of heuristic algorithm. It is not designed for a specific problem, but is a general heuristic algorithm framework. This type of method is more suitable for solving combinatorial optimization problems. In computing task offloading optimization scenarios, task offloading decision-making and resource allocation methods based on meta-heuristic algorithms can quickly obtain acceptable near-optimal solutions at low computational costs.

The work [96] used the greedy algorithm and discrete bat algorithm to explore the optimal decision of task offloading. This mechanism is aimed at the dynamic and mobile Internet of Vehicles computing environment, so it focuses on the important parameter of network communication hop. The mechanism works in a distributed manner and supports task offloading to MEC and other vehicles.

The study [97] proposed the concept of computational paths, thereby allocating computational tasks to multiple nodes on the computational path. The allocation process is based on a heuristic strategy and mainly considers the processing capabilities of edge devices, task deadline requirements, and dynamic changes in real-time workloads. In addition, this mechanism also incorporates user offloading references into decision optimization.

A heuristic based offloading method is proposed in Ref. [98] that uses an interior point algorithm (IPA) and Lagrangian multiplier (LM) to solve the optimization problem. This method optimizes task offloading while also taking into account reliability guarantees based on network fault tolerance.

For the cooperative application scenario of high-altitude platforms (HAPs) and unmanned aerial vehicles (UAVs), a layered aerial computing offloading framework for IoT is proposed in Ref. [99]. A matching game mechanism is introduced in this framework to select the highest priority UAV for each IoT device as the target execution node for task offloading. This mechanism supports many-to-one matching between IoT devices and UAVs.

An algorithm based on Lagrange Dual Decomposition (LDD) and a heuristic algorithm is proposed in Ref. [100], which solves the IoT task offloading optimization problem in MEC networks built based on low Earth orbit (LEO) satellite networks. The algorithm jointly optimizes task offloading decisions and allocation of limited available resources.

In [101], a GA based task offloading method was proposed for jointly optimization of UAV-server energy consumption and task execution latency. The method works by optimizing the trajectory of the UAV for providing the appropriate communication bandwidth and signal power for mission data transmission from the TD. Both battery-powered TDs and UAVs extended effective operating time. However, such algorithms will have difficulty in finding an acceptable approximate optimal solution quickly when the scale of MEC network is very large.

A multi-user MEC system using NB-IoT and taking into account the unique characteristics of NB-IoT compared to other wireless technologies is introduced in Ref. [102]. This research will model the NB-IoT system as a continuous-time MDP (CTMDP) model and propose a task offloading optimization method based on approximate dynamic programming (ADP). The algorithm allows IoT devices to make distributed offloading decisions and supports task scheduling and distribution among base stations.

A framework for joint task offloading, communication, and computing resource allocation for sequential tasks is proposed in Ref. [103]. Slow-fading and fast-fading channels are considered. The goal is to minimize energy consumption while ensuring task computation latency. The authors decompose the problem into a one-dimensional search and non-convex optimization problem of task offloading decision problems. Through mathematical processing, the non-convex problem is converted into a convex problem and solved using the Golden search method.

A low-complexity heuristic algorithm is provided in Ref. [104]. The principle is to adjust the scheduling strategy, minimum delay, and energy consumption by calculating the scheduling value and data transmission rate of IoT offloading tasks. This algorithm mainly considers energy consumption and time constraints for task completion.

In [105], the study focuses on the task offloading request from TDs and UAVs' intermediate relay scheme with dual constrains of QoE and battery limitation. The study proposes a HJPQ algorithm based on meta-heuristic method. The HJPQ included genetic algorithm and jointly consider task excuation delay, wireless network status, and the mobility of TDs. In addition, A DDPG-based task scheduling method was also implemented in this study. Experimental results show that HJPQ is better at guaranteeing QoE while optimizing latency and energy consumption. However, the proposed algorithm is difficult to support complex dependencies between multiple tasks.

The IoT task offloading mechanism in MEC based on heuristic and meta-heuristic algorithms has multiple advantages. First, heuristic algorithms usually exhibit fast calculation speed, allowing them to quickly generate solutions in IoT applications with high real-time requirements. Secondly, the implementation of these algorithms is relatively simple and does not require complex mathematical modeling and solving, thereby reducing system deployment and maintenance costs. Furthermore, they have broad applicability and are suitable for various IoT application scenarios regardless of the problem structure. In addition, the heuristic algorithm can quickly adapt to dynamic changes, such as device connection, disconnection, and data traffic fluctuations, and dynamically adjust resource allocation and task offloading. However, the limitation of heuristic and meta-heuristic algorithms is that they usually provide approximately optimal solutions rather than optimal solutions, which not be suitable in some applications that require high accuracy,

and the performance is greatly affected by the problem instance and algorithm parameters. Furthermore, they sometimes get stuck in locally optimal solutions and fail to find the global optimal solution and thus perform poorly in situations of high problem complexity. In actual applications, choosing an appropriate task-offloading method requires comprehensive consideration of application scenarios and performance requirements. Table 8 presented a side-by-side comparison of IoT task offloading mechanisms based on heuristics.

5.5. Game theory based offloading mechanisms

Game theory provides a framework to model and analyze how multiple entities make decisions and how their choices affect the overall system. In the context of task offloading, game theory is applied to scenarios where multiple agents decide whether to offload a task to other resources or execute it locally. The player's goal is to optimize his own goals, such as minimizing energy consumption or task execution time, while taking into account the actions and strategies of other players. Task offloading mechanisms based on game theory are particularly important in scenarios where multiple entities with conflicting interests interact, and the optimization of resource allocation depends on the behavior of multiple entities for all participants in the system.

While the work in Ref. [93] a game theory method for optimizing computation offloading strategies is proposed. The authors first established a system model in satellite edge computing, taking into account intermittent Earth-satellite communications caused by satellite orbits. Then, a computational offloading game model is proposed, in which each device selfishly chooses a strategy that minimizes its cost. The response time and energy consumption of tasks are calculated through queuing theory and these metrics are used to optimize performance. In order to find the Nash equilibrium of the game, an iterative algorithm is proposed.

In [94], the author provided a cooperative game algorithm based on cooperative offloading for single-task models and extended it

Table 8
A side-by-side comparison of offloading mechanisms based on meta-heuristic.

Ref.	Problems addressed	Technique	Network	Objective	Evaluation tools	Baseline	Advantages	Limitations
[96]	Task assignment	BAT	WAVE	Minimize latency	Python based implementation	Classic greedy algorithm	Multi-hop	Lack of MEC available resources to consider
[97]	Task assignment	Greedy winner selection	Undefined	Minimize latency, Maximize success rate	EdgeCloudSim, ElasticSim	Nearest, Selfish	Support dependencies between tasks	Lack of network feature support
[98]	Task assignment, resource allocation	Greedy, gradient decent	5G, WiFi, and ZigBee	Minimize switching latency, maximize QoS	Mininet	JONSSPE, SDENTO, JONSSPE	Probabilistic guarantees of offloading	Low resource utilization
[99]	Task assignment	Greedy, game theory	AAN	Maximize total IoT data computed	Matlab	Exhaustive searching	Support offloading to TD and MEC	Channel utilization is not considered
[100]	Task assignment, resource allocation	Greedy, lagrange dual decomposition	Low-earth orbit satellite networks	Minimize total delay	Matlab	Optimal exhaustive search	Low computational complexity	Load balancing is not considered
[101]	Task assignment, resource allocation	GA	MIMO	Minimize latency and energy consumption	Undefined	DQN	UAV trajectory optimization and battery life constraints	Slow convergence in large-scale MEC network scenarios
[102]	Task assignment	Linear value-function approximation, temporal-difference learning	NB-IoT	Minimize latency and energy consumption	Matlab	RR, QA, MUMTO	Simple, consider load balancing	Multi-objective optimization is not supported
[103]	Task assignment, resource allocation	Golden search method	NOMA	Maximize efficiency	Implementation	LARAC	Support task dependencies	Only for a special type of computation task in a MEC system
[104]	Task assignment	Greedy DAG graph calculation	Universal	Minimize latency and energy consumption	Implementation	PGBO, DEFO, PCDO, LOCOM	Support task dependencies	Does not support dependent subtasks with tight deadlines
[105]	Task assignment, resource allocation	GA	MIMO	minimize the overall cost, but also meet QoE requirement.	Matlab, Python	Random, DDPG	Lower complexity, faster convergence	Difficulty in dealing with dependencies between tasks

to a multi-task model algorithm. In the single-task model, the algorithm starts with all players joining a remote alliance and then iteratively performs alliance splitting until a stable split is reached. In the multitasking model, the algorithm additionally calculates the cumulative cost and remote incremental cost of each device and determines the optimal upload completion time. This algorithm realizes collaborative computing offloading of tasks, thereby improving the timeliness of task completion and reducing energy consumption.

A sub-gradient-based non-cooperative game model is presented in Ref. [95], which solves the task offloading problem in ultra-dense network environments. Taking into account the limited computing resources and dynamic needs of mobile users, the author proposes a Multi-objective Non-dominated Sorting Genetic Algorithm (MO-NSGA) based on Non-dominated Sorting Genetic Algorithm II (NSGA-II) to solve the problem of scheduling numerous task offloading requests in ultra-high-density networks. This mechanism adapts to the dynamic system environment and tries its best to reduce the energy consumption for data transmission while ensuring low-latency responses to task requests.

A computation offloading strategy based on a two-stage latent game is proposed in Ref. [96], which optimizes resource allocation strategy while taking into account the priorities of tasks and users in edge-enabled Wireless Body Area Network (WBAN). The original problem is reduced to a non-cooperative game process based on the underlying game model. Each task in the game attempts to maximize its utility and increase the effectiveness of the overall system. Two different policies in the policy space are resource allocation and offloading decisions. The first stage of the algorithm focuses on resource allocation and offloading decisions within the WBAN, while the second stage moves the game space to the MEC server. Tasks from different WBANs start the game in the second phase and obtain computing resources according to their utility.

In [97], the task offloading is modeled as a non-cooperative game and uses Nash equilibrium as the basis for decision-making. In order to solve the game problem, a distributed iterative algorithm is designed. This algorithm uses the Proximal Decomposition Algorithm as a regularization technique to solve the game problem through iterative convergence. The objective function of the algorithm includes two aspects: local computing cost and transmission cost. By minimizing the objective function, each MEC-BS can achieve its own optimal task offloading decision.

The IoT task offloading mechanism using game theory-based algorithms in MEC has unique advantages. First, they allow negotiation and gaming between devices to obtain maximum personal benefit, thereby promoting efficient allocation of resources. Secondly, this mechanism can achieve a fair allocation of resources among multiple devices, ensuring that all devices can obtain reasonable service quality. In addition, algorithms based on game theory usually have high flexibility and can adapt to various IoT application scenarios. However, these algorithms also have some limitations, including higher computational complexity, the need for cooperation from game participants, greater demand for information, and the possibility of introducing additional communication overhead. Table 9 presented a side-by-side comparison of IoT task offloading mechanisms based on game theory.

5.6. AI based offloading mechanisms

Artificial intelligence-based task offloading mechanism is a strategy that uses artificial intelligence technologies such as machine learning to make intelligent decisions about the allocation of computing tasks in a distributed computing environment. These mechanisms utilize artificial intelligence algorithms to analyze various factors, including network conditions, device capabilities, and user preferences, to determine the best offload strategy. The main advantage of AI-based offloading mechanisms is their ability to adapt to changing conditions and uncertainties, providing real-time and context-aware decision-making. Moreover, large-scale networks and massive devices often require automated and intelligent decision-making processes. Table 10 presented a side-by-side comparison of IoT task offloading mechanisms based on AI.

Table 9
A side-by-side comparison of offloading mechanisms based on game theory.

Ref.	Problems addressed	Technique	Network	Objective	Evaluation tools	Baseline	Advantages	Limitations
[93]	Task assignment	Game theory	Satellite	Minimize average cost	Iridium constellation	Unknown	Nash equilibrium decisions	Lack of global optimization
[94]	Task assignment	Game theory	5G	Minimize the overall cost of all TDs	Test-bed	Local executions	Nash-stable solution with convergence guarantee	Not suitable for high-density large-scale networks
[95]	Task assignment, resource allocation	Game theory	NOMA	Minimize energy consumption and latency	Matlab	Yalmip, ROGS, HOBS	Good convergence property	Ignores the impact of communication noise
[96]	Task assignment	Game theory	WBANs, RAN	Maximize system utility, minimize delay and energy consumption	Python based implementation	All local, all offload	Acceptable framework	Not compared with other algorithms
[97]	Task assignment	Game theory	RAN	Minimize latency	Matlab	IPA, OPEN	Support multiple scenarios	Not optimized energy consumption

Table 10
a side-by-side comparison of offloading mechanisms based on AI.

Ref.	Problems addressed	Technique	Network	Objective	Evaluation tools	Baseline	Advantages	Limitations
[98]	Task assignment	SVM	RAN	Minimize service delay, computation time, and service lag	OPNET	IHRA, COM, PCOA	Acceptable framework	Difficult to implement on large-scale training samples
[99]	Resource allocation	CNN, RNN, LSTM	5G	Maximize accuracy and detection rate	Python based implementation	RP, DE	Predict task offloading timeliness	Lack of comprehensive consideration of network parameters
[100]	Task assignment, resource allocation	DNN	WPT	Maximize efficiency of the edge server	Python based implementation	DROO, C D, OP, KNN	Support task offloading between TDs	If the MDs have a high mobility, the framework could be difficult to converge.
[101]	Task assignment, resource allocation	DNN	RAN	Maximize system utility and bandwidth allocation for each MD	Python based implementation	Local-Only, Edge-Only, Central-Only, Local and Central	Acceptable framework	Lack of comprehensive consideration of network parameters
[102]	Task assignment, resource allocation	DNN	OFDMA, SDN	Minimize delay and energy consumption	Python based implementation	GA, BNB, DQN	Good tradeoff between complexity and utility performance	The multi-tasking offloading scenario on TD is not considered
[103]	Resource allocation	DNN	WPT, NOMA, TDMA	Maximize computation rate	PyTorch based implementation	OFS + CVX, local-Only, edge-only	Acceptable framework	Ignores the mobility of TD
[104]	Task assignment	DNN	RAN	Minimize delay, energy consumption, and offloading overhead	Matlab	TOT, ROT, DOT, EEDOT, DIOT	Acceptable framework	Ignores the mobility of TDs
[105]	Resource allocation	SARSA, fuzzy logic	5G	Maximize service time, minimize task failure rate	EdgeCloudSim	Util, owb, fu-comp, hybrid	Taking into account system overhead	Only for specific application scenarios
[106]	Task assignment	Q-Learning, game theory	RAN	Maximize system utility	Matlab	Local-lony, Edge-only, Random assignment, BR	does not require environment model and payment information of other devices	Ignores the mobility of TDs
[107]	Task assignment, resource allocation	DQN, ACS, PSO	RAN	Maximize system utility	Testbed	BNEA, LCPSO, MOACS	Support multi-hop network	System overload is not considered
[108]	Task assignment	DQN, PDS-learning	RAN	Minimize delay and energy consumption	Keras and TensorFlow	DECENT, OEOFG, POOIE, OCODR	Supports both Online and Offline Offloading optimization	Ignoring the dynamic scalability of MEC computing power
[109]	Task assignment	DQN	RAN	Minimize delay and energy consumption	TensorFlow and Cooja	DTODR, ODRL, JMOR, ECCO, JSOE	Battery lifetime optimization	Task offloading between TDs is not supported
[110]	Task assignment	DQN	RAN	Minimize energy consumption	Tensor flow	Q-learning, FRAA	Both delay-tolerant and non-delay tolerant scenarios are considered	Collaborative offloading of multiple MEC nodes is not supported
[111]	Resource allocation	DQN	6G	Minimize consumption overheads of system energy and latency	TensorFlow	Traditional DQN Algorithm, FL, Random Selection Scheme	Better convergence	TD movement at the edge of the covered area is not considered
[112]	Task assignment, resource allocation	DQN	RAN	Maximize the total utility	Implementation	VES, FES	Support task offloading between TDs	Energy consumption optimization is ignored

(*continued on next page*)

Table 11
Parameters supported by the iot task offloading mechanism in MEC.

Work	Size of task	Task dependency	Dead-line	Capacity of TD	Capacity of MEC	*Trans*-mission delay	Channel bandwidth	Noise power	Location of TD	Network hop	Transmit power	Battery level	Pay-ment cost
[83]	✓		✓	✓	✓	✓	✓				✓		
[84]	✓		✓	✓	✓	✓	✓				✓	✓	
[85]	✓		✓	✓	✓	✓	✓	✓			✓		
[86]	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓	
[87]	✓		✓	✓	✓	✓	✓				✓		
[88]	✓		✓	✓	✓		✓	✓			✓		
[89]	✓		✓	✓	✓	✓	✓				✓		
[138]	✓		✓	✓	✓	✓	✓						
[139]	✓		✓	✓	✓	✓	✓	✓			✓		
[140]	✓		✓	✓	✓				✓				✓
[141]	✓		✓	✓	✓	✓	✓						
[142]	✓			✓	✓	✓	✓	✓			✓		
[143]	✓					✓	✓		✓		✓		
[144]	✓		✓	✓	✓	✓	✓	✓			✓	✓	
[145]	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓	
[146]	✓		✓	✓	✓		✓				✓	✓	
[147]	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓	
[96]	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓		
[97]	✓		✓	✓	✓	✓							
[90]	✓		✓	✓	✓		✓				✓		✓
[91]	✓		✓	✓	✓		✓	✓					
[98]	✓		✓	✓	✓		✓				✓		✓
[99]	✓		✓	✓	✓	✓	✓		✓		✓	✓	
[100]	✓		✓	✓	✓				✓				
[102]	✓			✓							✓		
[103]	✓	✓	✓			✓	✓	✓			✓		
[104]	✓	✓	✓	✓	✓		✓	✓			✓		
[92]	✓		✓	✓	✓	✓	✓	✓			✓		
[93]	✓		✓	✓	✓	✓	✓		✓	✓			✓
[94]	✓		✓	✓	✓	✓	✓	✓			✓		
[107]	✓			✓	✓	✓	✓				✓		
[108]	✓		✓	✓	✓	✓	✓				✓		
[109]	✓		✓	✓	✓		✓				✓		
[110]	✓		✓	✓	✓		✓						
[111]	✓		✓	✓	✓		✓						
[112]	✓		✓	✓	✓		✓						
[113]	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓	
[114]	✓		✓	✓	✓	✓	✓						
[115]	✓		✓	✓	✓	✓	✓	✓		✓	✓	✓	
[116]	✓	✓	✓	✓	✓	✓	✓	✓			✓		
[117]	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓	
[118]	✓		✓	✓	✓	✓	✓						
[119]	✓		✓	✓	✓	✓	✓	✓			✓		
[120]	✓		✓	✓		✓	✓			✓	✓		
[121]	✓		✓	✓	✓	✓	✓	✓			✓	✓	
[122]	✓		✓	✓	✓	✓	✓		✓		✓	✓	
[123]	✓		✓	✓	✓	✓	✓	✓			✓		
[124]	✓		✓	✓	✓			✓					

(*continued on next page*)

Table 11 (*continued*)

Work	Size of task	Task dependency	Dead-line	Capacity of TD	Capacity of MEC	*Trans*-mission delay	Channel bandwidth	Noise power	Location of TD	Network hop	Transmit power	Battery level	Pay-ment cost
[125]	✓		✓	✓	✓	✓	✓	✓	✓		✓		
[126]	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓	
[127]	✓		✓	✓	✓			✓					
[128]	✓		✓	✓	✓	✓	✓						
[129]	✓		✓	✓	✓	✓	✓		✓		✓		
[130]	✓	✓	✓	✓	✓	✓	✓	✓			✓		
[131]	✓		✓	✓	✓	√	✓	✓			✓	✓	
[132]	✓		✓	✓	✓	✓	✓				✓		
[133]	✓		✓	✓	✓	✓	✓				✓		
[134]	✓		✓	✓	✓	✓	✓	✓			✓		
[135]	✓		✓		✓	✓	✓	✓			✓		
[136]	✓		✓	✓	✓	✓	✓						
[137]	✓		✓	✓	✓	✓	✓				✓		
[148]	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓	
[149]	✓		✓	✓	✓	✓	✓	✓			✓		
[150]	✓		✓	✓	✓	✓	✓	✓			✓		
[151]	✓		✓	✓	✓		✓	✓	✓		✓		
[152]	✓		✓	✓	✓	✓	✓				✓	✓	
[153]	✓		✓	✓	✓	✓	✓	✓			✓		
[154]	✓		✓	✓	✓	✓	✓	✓	✓		✓		
[155]	✓		✓	✓	✓	✓	✓	✓		✓			
[156]	✓		✓	✓	✓	✓	✓	✓			✓		
[157]	✓		✓	✓	✓	✓	✓				✓	✓	
[158]	✓		✓	✓	✓	✓	✓	✓			✓	✓	
[159]	✓		✓	✓	✓		✓	✓			✓		
[160]	✓		✓	✓	✓	✓	✓		✓		✓		

Table 10 (*continued*)

Ref.	Problems addressed	Technique	Network	Objective	Evaluation tools	Baseline	Advantages	Limitations
[113]	Task assignment, resource allocation	DQN	RAN, WPT	Maximize computation time and execution latency	Testbed	Coordinate Descent, Linear Relaxation	Support task offloading between TDs	The mobility of the TDs would make it harder to converge
[114]	Task assignment, resource allocation	DQN	RAN	Maximize number of tasks completed on time, minimizing energy consumption	PyTorch based implementation	Greedy	Maximize long-term accumulated rewards instead of a one-time step	Energy consumption optimization is not considered
[115]	Task assignment, resource allocation	DQN	RAN	Minimize offloading cost	Matlab	OOS, DROS, GOS, ROS, EOS, COS	Supports offloading tasks to edge and cloud at the same time	Task offloading between TDs is not supported
[116]	Resource allocation	DQN	NB-IoT	Minimize long-term average weighted sum of delay and power consumption	Implementation	QA, MUMTO, Neural-ICO	tradeoff in minimizing the weighted sum of the delay and power consumption	Collaborative offloading of multiple MEC nodes is not supported
[117]	Task assignment, resource allocation	DQN	NOMA	Minimize the total energy consumption	TensorFlow, Keras	FDMA	joint optimization of the computation offloading, NOMA transmission, and computation resource allocation	Only supports static channel scenario
[118]	Task assignment, resource allocation	DQN	RAN	Minimize delay and energy consumption	Testbed	NO, EO, CMDP	Differentiated levels of task privacy requirements	Only for specific application scenarios
[119]	Task assignment, resource allocation	DQN, CNN, Lyapunov	LOS, NLOS	Minimize delay and energy consumption	Python based implementation	Random policy, greedy	tradeoff between edge preprocessing and network transmission	Ignores the mobility of TDs
[120]	Task assignment, resource allocation	DQN	Wi-Fi, LTE	Minimize delay and energy consumption	Testbed	Greedy, DQL-non-DP	Self-learning of wireless channel characteristics	Does not support multiple UAV scenarios
[121]	Resource allocation	DDQN, DDPG	RAN	Minimize computational cost	Python based implementation	OS, ES, LC	Acceptable framework	Only for specific application scenarios
[122]	Task assignment, resource allocation	DDQN	RAN	Minimize offloading and smart contract costs	TensorFlow, Testbed	DRLO, EO, CO	High security of offloaded data	Lack of consideration for optimizing offloading between TDs
[123]	Task assignment, resource allocation	PG	RAN	Minimize computation time	TensorFlow	Random, FIFO, SJF, PEFT, SAC	More robust approach	lack of energy consumption optimization
[124]	Task assignment, resource allocation	DDPG	5G	Minimize delay and energy consumption	PyTorch based implementation	Edge-only, DQN-based offloading, AC	Avoid noise interference	Assume that the network environment is stable and controllable
[125]	Task assignment	MADDPG	5G	Maximize utility of UAVs	TensorFlow	DDPG, A3C, Dueling-DQN	Support cluster-based multi-UAV network	Only for specific application scenarios
[126]	Task assignment, resource allocation	DDPG	RAN	Minimize total energy and delay	TensorFlow, tkinter	Greedy MECF, Optimal MECF, RRA, URA, DDPG, DDPG-PER	Load balancing among MEC nodes	Ignores the mobility of TDs
[127]	Task assignment	DDPG	MIMO	Minimize data buffer delay, energy consumption, bandwidth cost	Python based implementation	Dueling-DQN, DDQN, greedy policy	Taking into account the caching process during data offloading	Lack of optimization of system resource utilization

(*continued on next page*)

Table 10 (*continued*)

Ref.	Problems addressed	Technique	Network	Objective	Evaluation tools	Baseline	Advantages	Limitations
[128]	Task assignment, resource allocation	MADDPG	OFDMA	Minimize delay and energy consumption	TensorFlow	DDPG, Dueling-DQN, DQN, greedy	QoS guarantee	No joint consideration of UAV trajectory planning
[129]	Task assignment, resource allocation	MADDPG	RAN	Maximize service satisfaction for IoTDs	PyTorch based implementation	ACS_CS, RO_CS, DDPG-JAPORA	Support multi-UAV collaborative task offloading	No joint consideration of UAV trajectory planning
[130]	Task assignment	MAQDRL	Wifi	Minimize delay and energy consumption	TensorFlow	MAPPO, InDRL, MADDQN, MARAND	Acceptable lightweight framework	Collaborative offloading of multiple MEC nodes is not supported
[131]	Task assignment	Actor Critic	RAN	Minimize total task processing delay in a long-term period	Python based implementation	NN, AC,	Vehicles can select multiple MEC offloading nodes in parallel	Lack of energy consumption optimization
[132]	Task assignment, resource allocation	MADRL	NOMA	Minimize delay and energy consumption	PyTorch based implementation	DDPG, I-DDPG, Expert Algorithm	Strong robustness	High vector dimension in large-scale network application scenarios
[133]	Task assignment, resource allocation	Actor Critic	Satellite	Minimize total task delay	Test bed	Random, Greedy	Low computational complexity	Lack of joint optimization of computing and communication resources
[134]	Task assignment, resource allocation	Meta-RL	RAN	Minimize delay and energy consumption	TensorFlow	Optimal Exhaustive Search, Random Offloading, Greedy, PPO-Based DRL	Quickly adapt to changes in the network environment	Task offloading between TDs is not supported
[135]	Task assignment, resource allocation	DRL, FL	WPT	Minimize delay and energy consumption	PyTorch based implementation	Linear relaxation algorithm, CD	High accuracy of offloading action	Collaborative offloading of multiple MEC nodes is not supported
[136]	Task assignment, resource allocation	FL, DDPG	OFDMA	Minimize delay and energy consumption	PyTorch based implementation	Random Offload, Greedy, DQN, DDPG	Incorporates the protection of user data privacy	Task offloading between TDs is not supported
[137]	Task assignment	FL Q-learning	FDMA	Maximize accuracy	PyTorch based implementation	PQB-OS, POS-CSI	High effectiveness for obtaining the optimal offloading strategy	Ignoring the optimization of bandwidth and transmission energy consumption

5.6.1. Support vector machine

In [98] a distributed computing and resource-sharing task offloading framework is proposed, which is based on a Support Vector Machine (SVM) for resource-sharing and service allocation in edge computing. This framework classifies pipeline and distributed computing and adjusts the configuration of services oriented to offload tasks from IoT. Services are assigned to requesting users in the MEC network environment. Use SVM for task and pipeline instrumentation for decision-making and resource-oriented computing. Edge devices are responsible for checking service allocations and providing resource sharing to requesting users to reduce service latency and task computation time.

5.6.2. Neural Network

Compared with simple SVM, CNN, DNN, and LSTM are also used as supporting technologies for task offloading. A framework based on hybrid deep learning algorithms is proposed in Ref. [99] to solve the dynamic multi-task offloading problem in IIoT networks. The framework uses a combination of CNN and LSTM algorithms to learn the spatiotemporal characteristics of tasks, and then outputs task offloading decisions and corresponding resource allocations based on conditions such as time-sensitivity constraints of the task.

In [100], a parallel offloading framework is proposed that uses deep neural networks as parallel offloading executors to generate offloading actions. This framework decomposes the optimization problem into offloading decision sub-problems and resource

allocation sub-problems. It automatically improves its action generation strategy based on different wireless fading conditions. In addition, the impact of wireless power transfer on wireless IoT task offloading decisions is also discussed. A DNN-based hybrid offloading model for MCC and MEC is proposed in Ref. [101]. The algorithm utilizes multiple parallel DNNs to generate optimal offloading decisions. The algorithm requires prior training of a DNN model using a dataset containing workloads and corresponding offloading decisions. Therefore, the adaptability of this algorithm is limited. A computational offloading and resource allocation algorithm based on distributed deep learning is introduced in Ref. [102], which utilizes multiple parallel DNNs to generate optimal offloading decisions and resource scheduling. In Ref. [103], the researcher uses the experience replay technique to train DNN, randomly selecting a batch of training samples from memory. In addition, this method takes into account the application scenario of wireless power transmission to power IoT nodes and considers the joint optimization between optimizing wireless power supply and wireless communication. In Ref. [104] the author also proposed a task offloading decision-making method based on DNN to reduce the delay of task execution and the battery consumption of TD. However, this method takes into account the type of distinction of tasks, thereby enabling finer-grained task offloading scheduling and optimization.

5.6.3. Q-learning

In [105] the author proposed a task-offloading algorithm that combines fuzzy logic and SARSA reinforcement learning. The algorithm determines the MEC computing node used to process the task by considering the network environment and mobile access network parameters. It defines the communication model and calculation model and calculates the transmission time and calculation time of the task. The algorithm solves optimization problems and makes task offloading decisions by minimizing latency and balancing load among MEC nodes. The algorithm uses a SARS-based reinforcement learning algorithm to iteratively update the Q value until it converges to the optimal Q value. The authors aim to optimize service times and task failure rates, especially when the system is overloaded. A system model for offloading IoT tasks to MEC networks is established in Ref. [106]. This model considers the available computing resources of TD, the available resources of each computing node of MEC, and multiple indicators related to wireless network transmission. This study considers the interference of wireless channels and multi-user computing offloading scenarios in dynamic environments and proposes an evolutionary game model combined with reinforcement learning to optimize IoT task offloading.

5.6.4. Deep Q-learning

In [107], the author designed a two-layer intelligent optimization algorithm based on DRL to solve the joint optimization problem of IoT task offloading and resource allocation. In this study, a virtual backbone network architecture was constructed based on available MEC resources and based on DRL, a search was performed from TD to MEC Compute the best path between nodes. The IoT task offloading problem is modeled as a Markov decision process (MDP) in Ref. [108]. Information such as the number of MEC computing nodes, task attributes, network status, and number of TDs are used for DRL learning. This scheme can be used to automatically optimize offline and online task offloading strategies. In Ref. [109], the author uses Q-tables to learn offline task offloading strategies and utilizes CNN to accelerate the learning process. This study adopted a transfer learning method, which improved the offloading efficiency. A study [110] modeled the problem as a Markov decision process (MDP) and solved it using the DRL algorithm. The state space takes into account the network environment, wireless communication resources, channel state changes, etc. This study lacks attention to the availability status of computing resources. A work [111] further proposed research on task offloading optimization for sixth-generation (6G) communication technology and introduced collective reinforcement learning methods for resource allocation and network optimization.

In the scenario studied by Ref. [112], the on-board computer has relatively sufficient computing resources. Therefore, vehicles are treated as computing nodes in the MEC network. Computational tasks from the TD or vehicle can be offloaded to other vehicles or fixed MEC servers. Since the offloading decision-making problem in this scenario is not a convex optimization problem, the author proposed a method based on reinforcement learning to dynamically adjust the task offloading decision and resource allocation strategy. In Ref. [113] an online task offloading algorithm based on deep reinforcement learning is proposed to optimize computing task offloading in large-scale networks. The algorithm learns from past task-offloading experiences through reinforcement learning and improves its task-offloading actions through DNN. Order-preserving quantization and adaptive parameter setting methods are used to achieve fast algorithm convergence. A work [114] proposed an end-to-end DRL algorithm to select the best edge server for offloading and allocate appropriate computing resources. The algorithm learns optimal strategies through interaction with the MEC computing environment to maximize long-term utility.

A new deep imitation learning (DIL)-driven edge cloud computing offloading framework is proposed in Ref. [115], which aims to minimize costs in MEC task offloading environments through optimal behavioral cloning. The authors formalize the problem as a multi-label classification problem and use the generated optimal offloading actions to train the model in an offline manner. In Ref. [116] the researcher modeled IoT task offloading as an average reward continuous-time Markov decision process model under an infinite time perspective. The author implemented a distributed resource auction mechanism based on deep reinforcement learning technology to coordinate TD's task offloading requests. A work [117] introduced new equations and constraints to transform the problem into multiple equivalent forms with convex properties. Afterward, the dual optimization problem was iteratively updated using deep learning methods. In Ref. [118], the author also optimizes task offloading decisions based on deep learning technology. However, this research focuses more on supporting task offloading cost optimization in blockchain application scenarios.

The work [119] studied the IoT task offloading application scenario in which UAVs act as MEC computing nodes. As an MEC base station, the UAV can continue to offload tasks to the cloud, thus forming a multi-layer offloading architecture. Different from other deep learning-based task-offloading mechanisms, this research focuses on UAV trajectory planning to improve the task-offloading

efficiency of IoT. In Ref. [120], the researcher also studied the task offloading optimization problem in MEC networks involving UAVs. However, this study optimizes the cumulative policy gradient calculation step in policy update. This research ensures the efficiency of task offloading while also taking into account the privacy protection of task offloading.

5.6.5. Double DQN

The work [121] focuses on task offloading and resource allocation issues in MEC-assisted Railway Internet of Things (RIoT) networks. The algorithm proposed by the authors handles mixed integer nonlinear programming (MINLP) problems by combining DDQN and DDPG. This method takes into account the allocation of wireless communication resources and the allocation optimization of MEC computing resources to minimize the weighted total cost of energy consumption and delay. The deep learning technology is used in Ref. [122] to optimize computing task offloading in task offloading scenarios for blockchain applications. Compared with the usual DRL-based task offloading mechanism, this research takes more into account the characteristics of task activities such as access control and authorization of blockchain business.

5.6.6. Deteministic policy-gradient

While the study in Ref. [123] a MEC resource scheduling and IoT task offloading mechanisms based on graph neural network are discussed. This research combines reinforcement learning with graph convolutional network technology to model the interaction between the agent and the environment by modeling it as a Markov decision process (MDP). The training process uses the Monte Carlo method. The work [124] adopted a DDPG (Deep Deterministic Policy Gradients) based approach to jointly schedule resource allocation and computation offloading in multiple UAV-assisted MECs. This method can optimize the distributed parallel task offloading activities of multiple TDs on multiple UAVs, thereby avoiding network transmission bottlenecks and reducing the overall task offloading delay. In Ref. [125] the application scenario of combining multi-agent deep reinforcement learning (MADRL) with unmanned aerial vehicle (UAV)-driven IoT networks is studied. This study utilized a Stackelberg game model and transformed it into an MDP model. After that, the author constructed a model-free multi-agent deep deterministic policy gradient (MADDPG) algorithm to find the optimal task offloading decision-making strategy. The work [126] also proposed a task offloading optimization model based on DDPG. The model is designed to be relatively simple, and it mainly considers task delay and energy consumption optimization issues in the IIoT network environment.

In [127] a deep reinforcement learning-based data offloading and renewable energy aware model is provided, and the goals are minimizing the total system cost of energy consumption, data transmission delay, and bandwidth allocation under time-varying channel states. A study [128] uses multi-agent deep reinforcement learning (MADRL) technology to optimize the allocation of limited computing resources in an MEC environment composed of multiple UAVs to minimize long-term computing costs in terms of energy and latency. The work [129] combined deep deterministic policy gradient (DDPG) with a cooperative multi-agent learning framework in their research, and used a centralized training and distributed execution solution to solve the non-stationary problem of the network environment.

5.6.7. Actor-critic based policy-gradient

In [130] a lightweight optimal task offloading algorithm called MAQDRL based on queuing theory is presented. The algorithm integrates queuing theory and uses multi-agent deep reinforcement learning to obtain optimal offloading strategies in dynamic and stochastic multi-user offloading environments. The decision-making network is trained in a centralized manner in a data center, while the vehicles perform task-offloading decisions in a distributed manner. The work [131] proposed a centralized training algorithm and distributed execution algorithm based on multi-agent DRL for IoV task offloading application scenarios [132]. proposed an offloading decision-making framework consisting of a hierarchical coalition of multi-agents, in which upper-level agents receive decision-making experience from lower-level agents and perform reinforcement learning. The Air-Ground Integrated Network (SAGIN) architecture is studied in Ref. [133], which uses UAVs for edge computing and satellites for cloud computing. A method based on deep reinforcement learning is proposed to optimize the dynamic offloading strategy in a dynamic SAGIN environment.

5.6.8. Meta reinforcement learning

The work [134] proposed a cache-assisted collaborative task offloading and resource allocation mechanism that can achieve collaboration and resource sharing between multiple edge nodes and mobile devices. This mechanism takes cache status into account when performing offloading, allowing tasks offloaded to edge servers to obtain raw data and calculation results directly from existing caches. This reduces the overhead caused by redundant computation and transmission for repeated tasks. The author also established multi-dimensional indicators based on the concept of quality of experience (QoE) and constructed a QoE-aware utility function that considers subjective user preferences, objective execution costs, task cache status, task allocation status, and resource and network status, and achieved rapid Decision making and resource allocation.

5.6.9. Joint federated learning and reinforcement learning

Federated Learning (FL) is a distributed learning framework that improves decision-making efficiency by allowing IoT devices and MEC servers to jointly train a shared global model. The IoT task offloading algorithm proposed by Ref. [135] includes four main components: offloading action generation, offloading policy update, DNN model aggregation, and adaptive learning rate method. The algorithm generates offloading decisions for each TD based on distributed reinforcement learning, regularly updates the offloading strategy, aggregates DNN models from TDs, and adjusts the learning rate based on the performance of the algorithm. The goal is to adaptively allocate computing and communication resources in large-scale dynamic MEC scenarios. The task offloading algorithm

proposed by Ref. [136] combines federated learning to protect privacy and improve training performance. The actor network and critic network are each composed of multiple fully connected layers and use RELU as the activation function. The work [137] also proposed an edge-assisted federated learning framework for IIoT networks to alleviate the hysteresis effect of task offloading and improve the training efficiency of the decision-making system.

5.7. Parameters to be determined for all type of machanism

In the field of task offloading within MEC networks, a series of key parameters must be carefully determined to ensure the efficient execution of computing tasks. These parameters serve as a compass for optimizing offloading decisions and resource utilization. Basic considerations involve aspects, such as the characteristics of the task itself, processing power requirements, expected execution time, and timeout tolerance. This understanding forms the basis for discerning whether tasks are better suited for offloading to a more powerful MEC server or local processing.

5.7.1. Task load

The properties of the computing task itself are one of the basic parameters that need to be considered by the offloading mechanism. Such parameters are the main basis for various algorithms to judge whether a task can benefit from the offloading process and where to offload it.

The most basic parameter in the properties of the task itself is the size of the task, which is usually expressed in terms of the number of CPU cycles required to complete task execution. In addition, in some algorithms, the amount of data carried by the task will be additionally considered. In this study, all task offloading mechanisms support this parameter.

The task completion time is another important parameter. Task offloading is meaningless if its execution time is longer than the local execution time. In addition, some tasks contain both a desired execution deadline and a tolerable execution deadline.

For algorithms that support partial task offloading, they also need to obtain dependency constraints between task fragments and between tasks and tasks, and thus to avoid waiting conditions between offloaded tasks.

5.7.2. Computing capacity

The computational capacity is specific to TD and MEC. The computational capacity of TD is used to predict the time a task will take to execute locally. Similarly, the computational flux of the MEC is used to predict the time the task will take to offload.

5.7.3. Network status

Network state information is used to predict the time it will take for a task to be transmitted. This time, together with the task computation time, constitutes the total time cost of task offloading. Such parameters are mainly presented as delay and channel bandwidth.

For algorithms that support TD mobility, it is also necessary to obtain the location of the TD and the network hop that the task transmission needs to pass through. This information can support the algorithm to infer the cost of task transmission in a more fine-grained way.

5.7.4. Energy consumption

Energy consumption is mainly for TDs due to their limited battery capacity. The basic task energy consumption is inferred from the amount of computation of the task. In addition, wireless data transmission also consumes energy. Therefore, some task-offloading algorithms also support network state monitoring. Most of such state parameters are in the form of noise power, transmit power, and battery level of the TD.

5.7.5. Payment cost

Payment cost is a parameter that needs additional consideration for task offloading. It can be expressed as the cost of using MEC computational resources and the cost of task data transfer for TD. Such parameters are used as one of the reference conditions for multi-objective optimization in the decision-making process of task offloading. Table 11 shows the main parameters supported by the IoT task offloading mechanism in MEC.

6. Result and discussion

The results and analysis section of this review provide a comprehensive synthesis of the included literature to provide insights and answers to the research questions. It is structured to promote a clear understanding of the findings and their implications. As mentioned in the previous sections, the systematic approach adopted in the review process ensured the reliability and robustness of the analysis. Furthermore, we summarize the analytical reports of the research questions previously presented in Section IV, as follows:

6.1. AQ1: problems addressed

RQ1 aims to determine what the main problems are that the IoT task offloading mechanism tries to solve. According to the investigation, it was found that task assignment and resource allocation are the basic parts of the IoT task offloading strategy in MEC. These processes involve determining which tasks are performed locally on the device and which tasks are offloaded to remote

resources. Resource allocation involves the allocation of specific resources (e.g., computing power, storage, network bandwidth) to efficiently perform offloaded tasks.

Over 40 % of the literature surveyed in this survey focused on task allocation as the primary problem-solving goal [73,74,77,79,81,84–86,88,138–140,143,145,90,91,147,94,96–98,100,104,106,108–110,125,130,131,137]. The goal of resource allocation is addressed in approximately 10 % of the reviewed literature [99,103,105,111,116,121]. In addition, 48 % of the research work focuses on both task allocation and resource allocation issues [76,80,83,87,89,141,142,144,146,92,95,101,102,107,112–115,117–120,122–124,126–129,132–136,148,149]. The Distribution and overlay of IoT task offloading problems addressed in MEC are shown in Fig. 4.

6.2. AQ2: optimization objectives

The second research question (RQ2) to be answered refers to the goals of existing IoT task offloading mechanisms and algorithms to be optimized. Based on the above-mentioned reviewed papers, it was found that the optimization goals of task offloading are divided into three broad categories: Delay, Energy consumption, and resource utilization.

6.2.1. Offloading efficiency

The primary goal of optimizing the task-offloading mechanism is to improve the success rate of task-offloading and reduce costs as much as possible. The survey results show that performance metrics related to offloading efficiency are the most numerous [73,75,76,142,146,94,100–104,113,124,125,128].

6.2.2. Delay and Energy consumption

For many IoT applications, especially those that require real-time data processing and fast response (such as smart transportation, telemedicine, etc.), latency is a key indicator. The optimization goal is to ensure that the computing tasks of IoT devices can be processed in the shortest time to meet the real-time requirements of the application. Latency-related performance indicators can be presented in a variety of ways, including Response time, Completion time, and Makespan, etc. [89,139,92,105,109,123,124,132,134].

Another important goal of optimizing the IoT computing task offloading mechanism is to reduce energy consumption, thereby extending the life cycle of battery-powered devices. Energy consumption indicators are expressed in many studies as Device energy consumption, Total energy consumption, and Average energy efficiency, etc [140,143,116–118,125,136].

6.2.3. Resource utilization

While ensuring the efficiency of task offloading, the offloading mechanism also tries its best to reasonably allocate computing tasks to MEC servers to make full use of the computing, storage, and network resources of these servers. Balanced allocation of resources helps avoid resource waste and improve overall system performance. Fig. 5 shows the Optimization objectives of task offload scheduling in MEC [82,83,141,90,96,105,112].

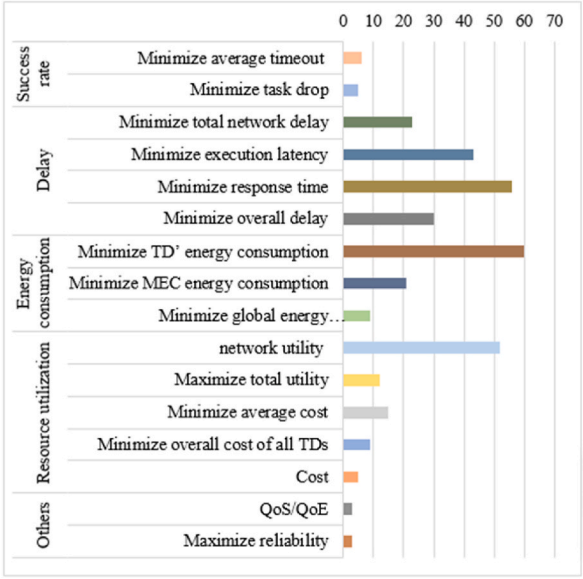


Fig. 5. Optimization objective of task offload scheduling in MEC.

6.3. AQ3: techniques

RQ3 intends to identify which techniques are used to implement the improved IoT task offloading algorithms and mechanisms. According to the proposed taxonomy, the technological evolution of IoT computing task-offloading mechanisms is moving from traditional methods to AI-based technologies in MEC networks. Traditional methods have the advantages of low computational complexity and accurate decision-making. However, AI-based offloading solutions are better suited to the IoT task offloading needs of large-scale complex application scenarios.

6.3.1. Traditional approach

Traditional approaches to computational task offloading in MEC network environments have unique advantages and disadvantages, and understanding this balance is critical in the context of modern, dynamic, and evolving network applications. On the positive side, traditional approaches are promoted for their simplicity, often relying on uncomplicated heuristics or simple task allocation rules. This simplicity means ease of implementation, making these methods available to a wide range of users and applications. In scenarios where rapid deployment and low complexity are critical, traditional approaches have clear advantages. Furthermore, traditional approaches typically have low computational and communication overhead compared to the complexity associated with optimization algorithms. This reduced overhead is especially beneficial in situations where minimal latency is critical, allowing for faster responses to real-time demands. Additionally, these methods provide a degree of predictability in task allocation and resource management, following predefined rules to provide users with consistent and expected performance. This predictability is invaluable in applications where stable and reliable operation is a fundamental requirement.

However, traditional methods are not without limitations. A significant drawback is their limited optimization capabilities. They often lack the sophistication to account for the full range of factors, such as dynamic network conditions, user preferences, or real-time changes. This limitation leads to suboptimal task allocation and thus reduced efficiency, which is a considerable problem in resource-limited MEC environments. Furthermore, traditional approaches have difficulty adapting to the dynamics and complexity of MEC network environments. They not be able to respond effectively to changing workloads, changing resource availability, or changing user needs. In environments where adaptability is critical, a lack of adaptability can lead to inefficiencies and degraded performance. In this study, traditional mathematical methods, Lyapunov optimization, heuristics, and game theory-based task offloading mechanisms are considered as traditional approaches.

6.3.2. AI-based approach

Artificial intelligence-based approaches to offloading computing tasks in MEC network environments constitute a dynamic field characterized by a subtle interplay between advantages and disadvantages. These aspects are critical to understanding the importance and potential impact of incorporating AI into the MEC framework.

In terms of advantages, artificial intelligence has demonstrated excellent optimization and adaptability, driven by the application of machine learning and deep learning algorithms. It is capable of dynamically assessing a wide range of factors, including real-time network conditions, user behavior, application-specific prerequisites, and edge device capabilities [150–152]. This adaptability enables smart task allocation to improve performance and optimize resource utilization. The strength of AI in predictive analytics leverages historical data and patterns to predict future network dynamics and user needs. This predictive capability enables proactive task allocation, ensuring tasks are assigned to the most appropriate edge resources. The result is reduced latency and an improved overall user experience, especially in applications where real-time responsiveness is critical. Furthermore, AI helps improve energy efficiency by coordinating task offloading strategies to reduce the energy consumption of edge devices. Tasks are judiciously allocated to edge servers that consume less power, thus extending the battery life of mobile devices – especially important in resource-limited environments and battery-powered devices.

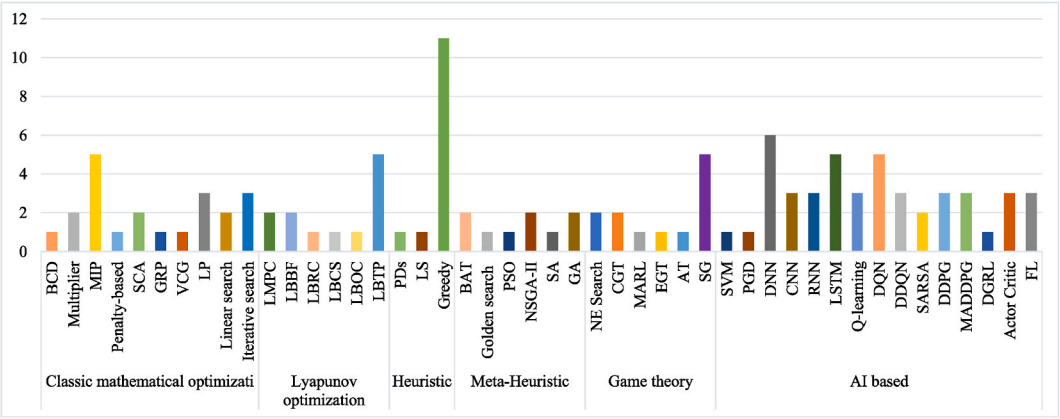


Fig. 6. The technical distribution of task offload scheduling mechanism in MEC.

Conversely, AI-based approaches also face several challenges. Their implementation is complex and often requires expertise in data science. The development and training of machine learning models are resource-intensive in terms of computing power and data, which creates barriers to adoption, especially in small-scale deployments. Scalability poses significant challenges given the volume of data and complexity of AI models. Large-scale deployment of AI-based methods requires significant computing resources and infrastructure, making scalability a key consideration. Artificial intelligence introduces computational overhead, which can lead to latency, which is a problem in scenarios where ultra-low latency is non-negotiable. Finally, deploying AI models in edge servers consumes resources and power. Balancing the resource consumption of AI against the benefits it provides, especially in resource-constrained MEC environments, remains a key consideration. Fig. 6 shows the technical distribution of the task offload scheduling mechanism in MEC.

In summary, Classic mathematical optimization ensures optimality through rigorous models but struggle with scalability and dynamic IoT environments. Lyapunov optimization, emphasizing stability, is adaptable but can be complex. Heuristic approaches offer speed and adaptability but yield suboptimal solutions. Game theory models strategic interactions, providing equilibrium solutions but can be computationally intensive. AI-based approaches have developed very rapidly since 2019. They are highly adaptable and can solve complex task-offloading optimization problems. Fig. 7 compares different task offloading approaches across different taxonomies of technology regarding adaptability, overhead, Multi-objective optimization capabilities, ability to support large-scale IoT and multiple parameters.

6.4. AQ4: network environments

The analysis of which network technologies are considered in IoT task offloading approaches in MEC is necessary to answer RQ4. Heterogeneous network environments play a crucial role in computing task offloading, affecting decisions about when, where, and how to offload tasks. The literature surveyed in this survey covers a wide range of wireless communication network technologies.

6.4.1. RAN

Radio Access Network (RAN) is a critical component of a mobile telecommunication system that connects user devices (such as smartphones, tablets, and IoT devices) to the core network and enables wireless communication. RAN manages the radio resources and communication between user devices and the core network, making it a fundamental part of cellular networks [153]. Some studies do not distinguish between multiple advanced wireless communication characteristics, but only consider several parameters of the basic RAN communication model [76,77,80,82,84,85,154–156].

6.4.2. LTE

A mobile communications standard built on RAN. Provides high-speed data transmission and better performance for 4G networks. A small number of task offloading optimization mechanisms take into account resource allocation and task scheduling based on LTE network characteristics [83,120].

6.4.3. 5G

5G, or the fifth generation of wireless technology, is the latest standard for cellular networks. It represents a significant leap forward in mobile communication technology, offering faster data speeds, lower latency, increased connectivity, and improved support for a wide range of applications and devices [157]. In the research on some task offloading mechanisms, full consideration has been given to using the characteristics of the 5G network to optimize the transmission process of task data [86,87,141,94,99,105,124,125,149,158,159]. In addition, there are also a small number of research attempts to explore the advanced features of future networks based on 6G

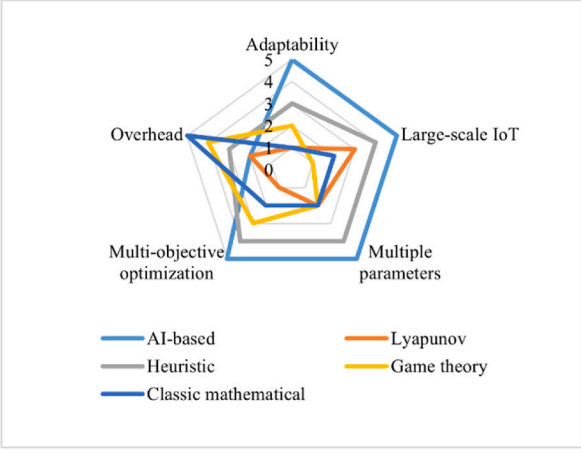


Fig. 7. Network environments of task offload scheduling in MEC.

to build IoT task offloading mechanisms in MEC [111].

6.4.4. TDMA and OFDMA

TDMA and OFDMA are multiple access techniques that divide communication resources differently. TDMA allocates time slots for different users, while OFDMA assigns sub-carriers in the frequency domain. Multiple access technology, is widely used in different mobile communication standards, including 4G LTE and 5G. In Refs. [75,103] the characteristics of TDMA were introduced into the communication model. The characteristics of OFDMA are used by several projects such as [73,102,128,136,137,160,161] to build communication models in task offloading. In Ref. [127] the author uses MIMO as the wireless communication environment for MEC networks.

6.4.5. NOMA

NOMA, or Non-Orthogonal Multiple Access, is a wireless communication technology that enables multiple users to share the same frequency, time, and code resources simultaneously within a cell in a cellular network [162].

In [78,146,95,117,163] the construction of the communication model fully takes into account the advantages of NOMA, so that the transmission of task data can be coordinated more precisely.

6.4.6. NB-IOT

Narrow-band IoT is a communication technology used to connect numerous low-power IoT devices, providing long-distance coverage and low power consumption. It is a communication technology specifically designed to connect low-power IoT devices. It can be used with modern communication networks such as 4G LTE and 5G to support IoT applications. However, there are relatively few studies on the task offloading mechanism of NB-IoT networks [145,116].

6.4.7. Wi-fi

A Wi-Fi network, also known as a wireless local area network (WLAN), is a type of local area network that uses radio waves to connect devices to the internet and each other without the need for physical wired connections. MEC supports Wi-Fi access mode to transmit data [164]. There have been some works studying the IoT task offloading mechanism based on Wi-Fi network [142,90,130,158,165].

6.4.8. WPT

Many TDs are powered by wireless methods due to the particularity of IoT. Therefore, WPT-based IoT task offloading methods have been considered in some studies [73,100,103,113,135,160,165]. The difficulty of such research lies in how to alternately transmit data and wirelessly supply power to TDs in a limited wireless spectrum.

6.4.9. SDN

Software-defined network is a network architecture and technology. Its core idea is to separate the network control plane and data plane to improve the flexibility, programmability, and automation of the network. The task offloading mechanism based on this network architecture can better obtain the status of network available resources and realize dynamic allocation of network resources, due to SDN's powerful management capabilities for the overall network [92,102].

6.4.10. Satellite

A satellite network is a telecommunications network that uses communication satellites to relay data signals between widely dispersed geographic areas on Earth. These networks provide global or regional coverage and are crucial for various applications,

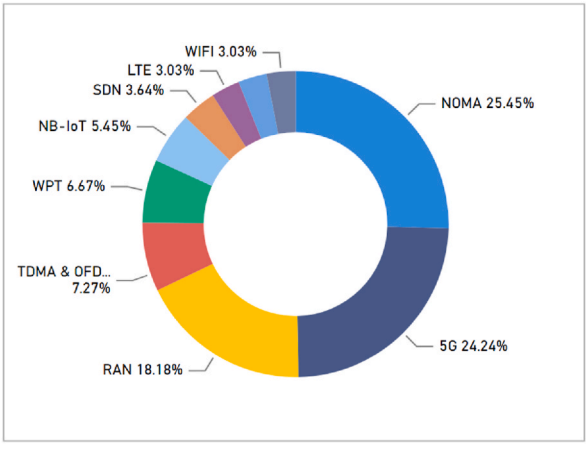


Fig. 8. Network environments of task offload scheduling in MEC.

including television broadcasting, internet access, and long-distance communication [166]. In Refs. [93,133,167] the task offloading mechanism fully takes into account the low bandwidth and high latency characteristics of the communication link. The network environments supported by task offloading in MEC are given in Fig. 8.

6.5. AQ5: architecture

Aiming to answer RQ5, the description, and architecture of IoT task offloading in MEC were extracted from the analyzed papers. Device-to-edge-and-cloud (D2EC) architecture in task offloading for computational tasks within a MEC network combines the processing capabilities of both edge servers and cloud resources to optimize task execution. This architecture provides flexibility and scalability, allowing users to select the most appropriate destination based on the specific requirements of their tasks. Fig. 9 presents the architectural distribution of task offloading scheduling in MEC.

6.5.1. TD to TD

TD to TD task offloading in a MEC network involves offloading a task from one user's device to another user's device for execution, rather than sending it to an edge server or cloudlet. This approach leverages the computational capabilities of nearby user devices to collaboratively execute tasks, providing various advantages and challenges [138,143].

6.5.2. TD to edge

Device-to-edge (D2E) task offloading in a Multi-Access Edge Computing (MEC) network involves offloading a task from a user's device to an edge server or cloudlet located at the network's edge. This approach allows tasks to be processed closer to the source of data, reducing latency and improving the overall performance of applications [73,74,76,77,79,80,83,84,86–89,144–147,91,94,97,100,102–104,106,107], [113,114,116–118,120,123,126–128,130,132,135,137,148,160,161,167].

6.5.3. TD to edge-cloud

Device-to-edge-cloud (D2EC) task offloading in a MEC network involves offloading a task from a user's device to a cloud infrastructure that is located at the network's edge. This approach combines the benefits of both edge computing and cloud computing to optimize task execution [81,85,139–142,90,92,95,96,98,99,101,108–112,115,119,121,122,124,125,129,131,133,134,136,149,156,168].

6.6. AQ6: offloading destination

RQ6 aims to determine whether algorithms and mechanisms are more inclined to offload IoT tasks to single or multiple servers in MEC. According to the results of the research analyzed, the offloading destination refers to the target resource or location to which a computational task is assigned for execution. Selecting the appropriate offloading destination is a crucial decision, as it significantly impacts factors like task execution time, energy consumption, and overall system performance. Offloading destinations can vary based on the nature of the computing environment and the specific objectives of the task offloading strategy. Fig. 10 shows the offloading destinations of the task offloading schedule in MEC.

6.6.1. Single server

In computational offloading within a MEC network, a single server as a task offloading destination refers to the scenario where a task from a user's device is offloaded and executed on a single edge server or cloudlet. This means that the entire computational load of the task is handled by a single server [73,80,84,88,89,143,145,146,100,103,104,106,110,113,116,120,130,135,137,148,149].

6.6.2. Multiple server

In computational offloading within a MEC network, multiple servers as task offloading destinations refer to the scenario where a

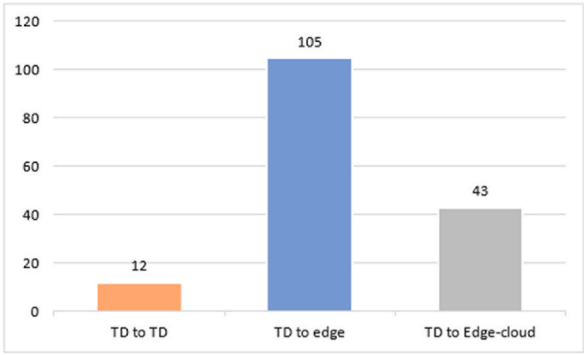


Fig. 9. The architecture of task offload scheduling in MEC.

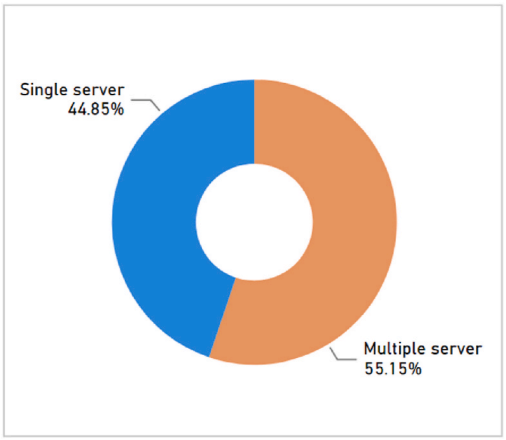


Fig. 10. Offloading destination of task offload scheduling in MEC.

task from a user's device is offloaded and executed on more than one edge server or cloudlet. This approach involves parallel execution of the task, with different servers sharing the computational load [74,77,81,85–87,138–142,144,90–92,147,94–99,101,102,105,107–109], [111,112,114,115,117–119,121–129,131–134,136,160,161,167,169].

6.7. AQ7: parameters

RQ7 aims to identify which parameters are used to support IoT task offloading algorithms in extracting state information from the MEC network environment that influences task offloading decisions. Parameters in computational task offloading algorithms are the configurable settings, variables, or inputs that allow researchers and practitioners to fine-tune the behavior and performance of task allocation and resource assignment strategies. These parameters influence how offloading decisions are made and the overall behavior of the algorithm. Fig. 11 illustrates the main parameters of the task offloading scheduling algorithm in MEC.

6.7.1. Size of tasks

The parameter size of a computational task in the context of computational offloading in a MEC network refers to the amount of data or the size of the task's parameters that need to be transferred between the user's device and the edge server or cloudlet for processing. The parameter size is a critical factor in determining whether offloading a task to the edge is practical and efficient. Almost every job considers this parameter.

6.7.2. Task dependency

Task dependency in computational offloading within a MEC network refers to the relationships and dependencies that exist between different computational tasks when determining whether and how they can be offloaded to edge servers for execution [146,170–176]. Task dependency plays a crucial role in optimizing the offloading process and ensuring that tasks are executed correctly and efficiently. Some task offloading solutions based on different technologies support dependency constraints between tasks [146,147,103,104,117,154,167,177–180]. However, some methods to support dependent task offloading do not consider load balancing

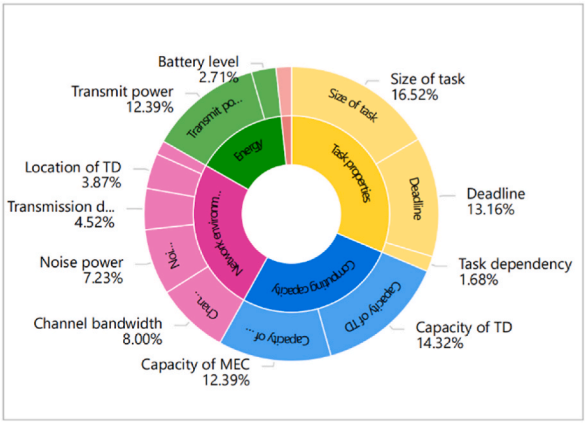


Fig. 11. Parameters of task offload scheduling in MEC.

[181–184] and heterogeneous network support for cloud-edge structures [185]. In addition, some studies only considered single-objective optimization [186–189].

6.7.3. Dead line

The deadline of a task in computational offloading within a MEC network refers to the maximum allowed time within which the task must be completed or delivered after it is offloaded to an edge server or cloudlet. Meeting task deadlines is critical, especially for real-time and time-sensitive applications, to ensure that results are available within the required timeframe. Almost every job considers this parameter.

6.7.4. Capacity of TD and MEC

The computational capacity of a terminal device is a pivotal factor within the realm of computational offloading in MEC networks. It encompasses the device's ability to perform computational tasks efficiently and includes aspects such as processing power, memory capacity, GPU capabilities, and latency sensitivity.

The computing capacity of the MEC server is one of the key parameters in the IoT task offloading mechanism. It represents the computing resources and processing power available on the server. This parameter is usually expressed in MIPS terms.

6.7.5. Transmission delay

The transmission delay of a task in computational offloading within a MEC network refers to the time it takes for the data associated with a task to be transmitted from the user's device to the edge server or cloudlet where the task will be executed. Transmission delay is a critical factor in the overall latency of task execution.

6.7.6. Channel bandwidth

The channel bandwidth in computational offloading within a MEC network refers to the amount of available frequency spectrum or bandwidth on the wireless communication channel used to transmit data between the user's device and the edge server or cloudlet where a task will be executed. The channel bandwidth is a critical factor that influences the data transfer rate, which, in turn, affects the transmission speed and overall task execution performance.

6.7.7. Location Of TD

The location of TDs in computational offloading within a MEC network refers to where and how the offloading of tasks is implemented in the network architecture. Task offloading involves creating multiple copies of a task for parallel execution on multiple edge servers or cloudlets to improve performance, ensure redundancy, or meet specific requirements [76,82,87,89,138,143,144,91,100,109,112,113,116,125,128,131,137,149,156,167,169].

6.7.8. Network hop

The network hop in computational offloading within a MEC network refers to the number of intermediate network devices or points that data associated with a task must traverse between the user's device and the edge server or cloudlet where the task will be executed. Each network device or point, such as routers, switches, and gateways, represents a "hop" in the network path. The network hop count is a critical factor that influences the overall latency and efficiency of data transmission in the context of task offloading [138,91,102,107,132,169].

6.7.9. Transmit power and NOISE POWER

The transmit power in computational offloading within a MEC network refers to the amount of power utilized by the user's device or the transmitting equipment to send data associated with a task to the edge server or cloudlet where the task will be executed. Transmit power is a critical factor that impacts the efficiency and reliability of data transmission in MEC network task offloading.

The noise power in computational offloading within a MEC network refers to the presence of unwanted or random electrical signals and interference in the wireless communication channel used to transmit data between the user's device and the edge server or cloudlet where a task will be executed. Noise power is a critical factor that affects the quality of the communication channel and can impact the reliability and performance of data transmission [78,81,84,86,87,89,138,90,146,147,92,95,100,103,104,106,108,111–114,117,121,122,126–128,130–133,135,136,167].

6.7.10. Battery level

The device's battery level in computational offloading within a MEC network refers to the remaining charge or energy capacity of the user's device, such as a mobile phone, IoT device, or sensor, which initiates the task offloading process. The device's battery level is a critical factor that influences offloading decisions and strategies to optimize the use of available energy [74,76,85–89,143,100,102,104,108,109,113,118,125,129,134,135,168,177,180,190].

6.7.11. Payment cost

The payment cost in computational offloading within a MEC network refers to the financial or monetary expenses associated with offloading a task from the user's device to an edge server or cloudlet for execution. This cost vary based on various factors and can impact the decision-making process for task offloading [82,140,142,91,93,190–193].

6.8. AQ8: evaluation METHODS

The RQ8 is to be answered according to the evaluation scheme stated in the literature surveyed. Evaluation methods in computational task offloading are techniques used to assess the performance, effectiveness, and efficiency of task-offloading strategies and algorithms in distributed computing environments. These methods help researchers and practitioners understand how well a particular task allocation and resource assignment approach meets the defined objectives. Fig. 12 shows the evaluation method of the task offloading mechanism in MEC.

6.8.1. Testbed

In order to evaluate the ability of the offloading mechanism, experimental environments have been built based on physical equipment in some studies. This type of experimental bed includes TD, MEC server, and network communication equipment [74,78,141,94,107,113,118,120,122,133,149].

6.8.2. Simulator

MATLAB (Matrix Laboratory) is a high-level programming and simulation environment widely used in performance evaluation and modeling for various technologies, including IoT (Internet of Things) task offloading mechanisms [194]. Judging from the survey results, Matlab is the main tool used for testing IoT task offloading algorithms and mechanisms. Researchers simulate TDs, MEC servers and network communication environments in Matlab, and load data sets to evaluate task offloading mechanisms or algorithms [75–77,79–81,84,86–89,143–145,95,97,104,106,115,165].

EdgeCloudSim is a simulation framework designed for evaluating the performance of IoT (Internet of Things) task offloading mechanisms in edge and cloud computing environments [195]. In Refs. [139,105] is used to build MEC infrastructure.

ElasticSim is a versatile performance evaluation tool designed for assessing IoT task offloading mechanisms. It provides a flexible simulation environment that accommodates various scenarios, allowing researchers and engineers to model and analyze task distribution strategies among edge and cloud resources [196]. In Ref. [139], the experimental environment used to evaluate task offloading algorithms is built on ElasticSim.

Mininet is an open-source network emulation tool that provides a lightweight and scalable platform for evaluating IoT (Internet of Things) task offloading mechanisms. It allows users to create virtual networks, mimicking real-world network topologies, and emulate various IoT device interactions and communications [197]. In Ref. [142] the author built the infrastructure of the IoT network based on Mininet.

NS-3, or Network Simulator 3, is a popular open-source simulation tool designed for performance evaluation in the context of IoT (Internet of Things) task offloading mechanisms [198]. In Ref. [91] NS-3 is used to build a network communication simulation environment for IoT and MEC.

OPNET (Optimized Network Engineering Tool) is a comprehensive and widely used simulation and modeling tool for assessing the performance of IoT task offloading mechanisms [199]. In Ref. [98] The author built a complex IoT deployment environment based on OPNET.

GT-ITM (Georgia Tech Internetwork Topology Models) is a simulation tool that focuses on evaluating the performance of IoT task offloading mechanisms by modeling network topologies. It offers a lightweight and scalable environment for simulating network scenarios [200]. In Ref. [90] GT-ITM is used to simulate various application scenarios of IoT networks.

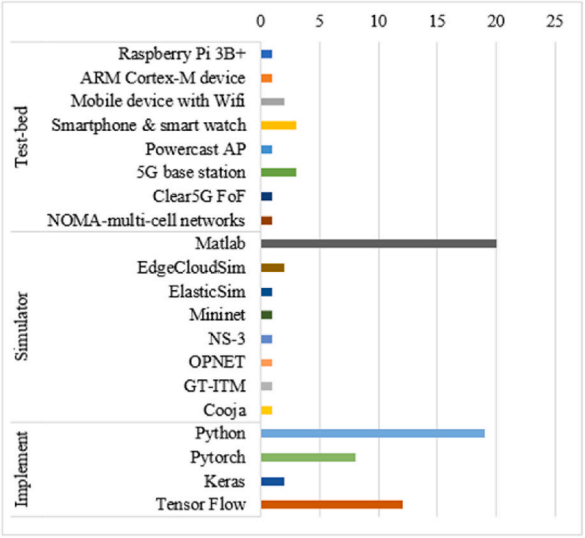


Fig. 12. Evaluation methods of task offload scheduling in MEC.

Cooja is an open-source simulator. It is a part of the Contiki operating system and provides a platform for modeling and simulating IoT networks and devices [201,202]. In Ref. [109] researchers use Cooja to assess task offloading strategies and network performance under various conditions.

6.8.3. Implement

In some studies, programming is used to simulate the IoT network environment to more accurately evaluate the application scenarios of IoT task offloading. Such implementations all use Python as the programming language.

[73,85,138,140,146,147,92,96,99–102,112,116,119,121,127,131,148]. In addition, Pytorch is introduced in some implementations to support machine learning algorithms involved in task offloading mechanisms [103,114,124,129,132,135–137]. Besides, in Refs. [108,117] Keras is also used to assist in the implementation of various AI-based task offloading algorithms. Relatively, Tensor Flow is the most widely used basic platform for task offloading AI algorithm implementation [83,108–111,117,123,125,126,128,130,134].

6.9. AQ9: baselines

The analysis of which baselines are considered in IoT task offloading approaches in MEC is necessary to answer RQ9. In computational task offloading, baselines serve as reference points or control strategies against which the performance of new or proposed task offloading mechanisms is compared. Baselines are essential for evaluating the effectiveness and efficiency of novel strategies and algorithms. They provide a standard for assessing whether a new approach outperforms existing, well-established methods. Upon investigation, it was found that the baseline chosen for evaluating the various algorithms and mechanisms was very decentralized. As a result, 80 different baselines are identified. Fig. 13 shows the evaluation of the top 10 baselines (occurrences at least twice) for IoT task offloading mechanisms.

6.10. AQ10: performance metrics

Aiming to answer RQ10, the performance metrics were extracted from the analyzed papers. Performance metrics in computational task offloading are quantitative measures used to assess the effectiveness and efficiency of task allocation and resource assignment strategies in distributed computing environments. These metrics help researchers evaluate how well an offloading mechanism performs and whether it meets the defined optimization objectives. Metrics have different designs in multiple studies, and they can be roughly classified into 5 categories. Fig. 14 shows the performance metrics of the task offloading mechanism in MEC. Based on the statistical analysis it can be visualized that the performance evaluation indexes are mainly distributed in energy consumption, time and task offloading efficiency.

6.10.1. Offload efficiency

Offload efficiency is a critical performance metric in the context of computational offloading within a MEC network. It measures how effectively and optimally tasks are offloaded from a user's device to edge servers, cloud resources, or other destinations. A high offload efficiency implies that the offloading process is achieving its intended objectives efficiently.

6.10.2. Time

Time-related performance metrics play a central role in evaluating the performance of computational offloading in MEC networks. The ability to minimize latency and optimize the timing aspects of task offloading, execution, and response is crucial for various real-time and latency-sensitive applications.

6.10.3. Energy consumption

Energy consumption is a key aspect of performance metrics for computational offloading within MEC networks. Efficient energy

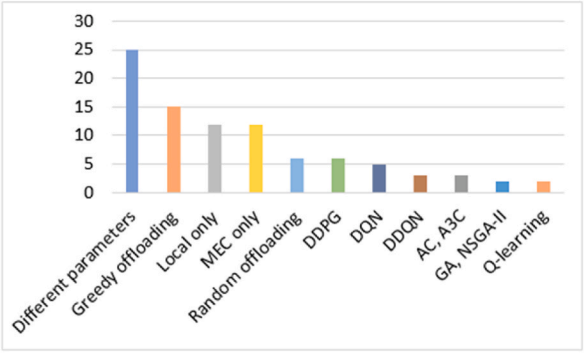


Fig. 13. Top 10 baselines of task offload scheduling in MEC.

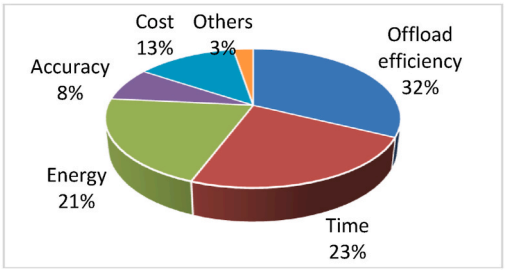


Fig. 14. Performance metrics of task offload scheduling in MEC.

management is critical to extending the battery life of user devices, lowering operating costs and reducing environmental impact. Efficient offloading decisions aim to minimize energy usage, especially during data transfer between the device and the offloading destination.

6.10.4. Accuracy

Real-time accuracy is especially important for applications such as autonomous vehicles and augmented reality, where tasks must be performed accurately within tight time constraints. This metric evaluates the speed and effectiveness of error handling and recovery mechanisms in identifying and correcting errors.

6.10.5. Others

In a few studies, task offloading mechanisms additionally take into account feedback from security requirements and offloading satisfaction.

For these performance metrics, it was found that some numerical expression methods with similar functions but different forms were used in different papers. In addition, some researchers have chosen different words to represent the same meaning. Fig. 15 shows the details of the performance metrics of the task offloading mechanism in MEC.

6.11. AQ11: dataset

RQ11 intends to identify which datasets are used to support the performance evaluation of the proposed IoT task offloading approach. According to the survey results, the data used in most of the work are specifically generated based on the experimental environment settings, and there is a lack of public datasets. Only a few works have evaluated and validated their proposed algorithms

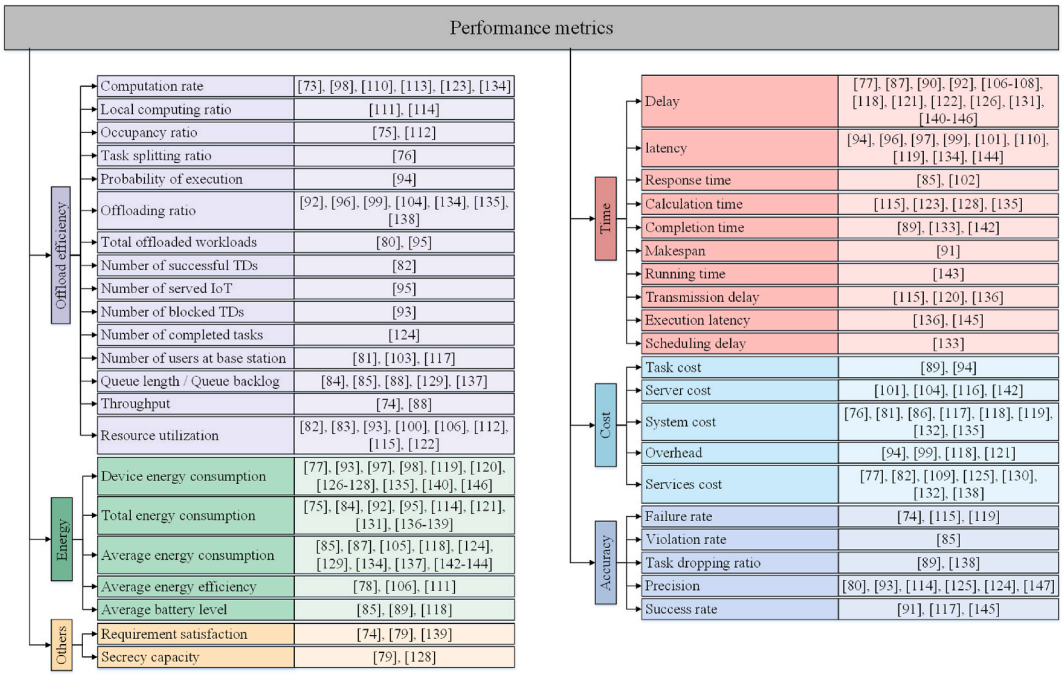


Fig. 15. Year wise publication of task offload scheduling in MEC.

and mechanisms using data collected from the real world. Some of these datasets provide computational tasks, while others provide trajectories for simulating TDs motion. Otherwise, most research works set their simulation parameters and generate special test data sets. Fig. 16 shows the data set used for the evaluation of the task offloading scheduling mechanism in MEC.

6.11.1. *Compute load data set*

DTLZ series functions have been widely used in the evaluation and benchmarking of multi-objective optimization algorithms and provided a standardized set of test cases to assess their efficiency and effectiveness in solving real-world multi-objective optimization problems [203,204]. Reference [77] evaluated and verified the effectiveness of the proposed task offloading mechanism based on this data set.

The ECG-ID database consists of 310 ECG signals obtained from different persons aged between 13 and 75 years [205,206]. This dataset is often used to generate keys for encryption algorithms due to the discrete nature of the data. The research work [81] generates random task offloading requests based on the ECG-ID dataset to test the performance of the proposed offloading mechanism.

The work [141] introduces five Chicago street video clips from the real world. This data set contains 9 types of target objects (people, handbags, backpacks, bicycles, cars, motorcycles, buses, trucks, and traffic lights) classification. In the evaluation experiment, the author used video content recognition as a computing task that needs to be offloaded to verify the effectiveness of the offloading mechanism.

A study [99] evaluated its work based on improved original data to verify its advantages in industrial computing task offloading scenarios. It contains a variety of service types with different timeliness requirements and can support task awareness in IIoT scenarios [207,208].

These TPC-H query workloads contain 22 different query models (i.e., the different DAG topologies) with 7 different query sizes [209,210]. Query workloads in Spark were selected as the computing task load to test the efficiency of the offloading mechanism in Ref. [123].

The MNIST dataset contained a training set of 60,000 samples and a test of 10,000 samples of handwritten digits from 0 to 9, which is widely used in the relative research work [211–216]. A study [137] takes the recognition process based on the MNIST dataset as a computational task to evaluate the proposed offloading mechanism.

6.11.2. *TD movement trajectory data set*

This type of data set provides TD's movement trajectory information and is used to generate distance changes between the TD and the base station, thereby affecting communication quality and delay factors.

A work [131] recorded the real vehicle trajectory of Rome city collected in 30 days. The time ranges from 2014/02/01 to 2014/03/02. The period is from 7 a.m. to 8 a.m. used to provide TD mobile trajectory simulation.

The infocom06 dataset was collected by the Cambridge Haggle projects. For the Infocom06 trace, it records 98 people's contact during the conference of IEEE Infocom 2006, and 40 IDs are randomly utilized to simulate the social relationship network [217,218]. A study [156] extracts the user movement process in this data set as the TD movement trajectory to evaluate the task offloading mechanism's ability to support TD mobility.

EUA is a public real-world dataset, which includes the geographical locations of 816 end-users and 125 base stations in Melbourne, Australia [219,220–222]. References [180,192,223,224] selected this data set as the distribution of MEC base stations and the activity trajectories of mobile devices, and then superimposed specific generated computing tasks to build a verification environment for the offloading mechanism.

Fig. 17 shows the distribution of the data set used for the evaluation of the task offloading scheduling mechanism in MEC.

7. Open issues and future research directions

In the journey through the existing literature, this review has identified several open issues and areas of continued inquiry within the field. The purpose of this section is to shine a light on these unresolved questions and highlight the frontiers of knowledge that

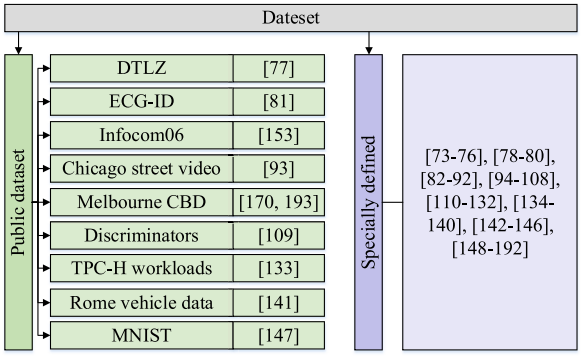


Fig. 16. Dataset of task offload scheduling in MEC.

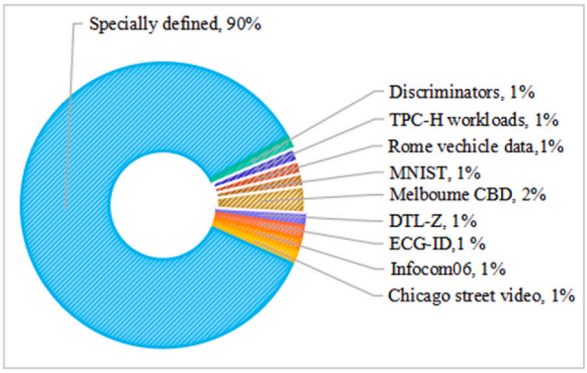


Fig. 17. Dataset of task offload scheduling in MEC.

warrant further exploration. By recognizing and discussing these open issues, we contribute to the ongoing dialogue and evolution of the field.

7.1. Joint decision-making by TD and MEC

Centralized and distributed offloading decision-making mechanisms each have advantages and disadvantages. The centralized offloading decision-making mechanism can grasp more global information and therefore can better achieve global optimization objects. The distributed decision-making mechanism has the advantage of being fully aware of the limitations of the terminal's local computing resources and has higher reliability. However, such methods lack awareness of global information. Moreover, large-scale distributed decision-making networks are difficult to converge quickly. This way, it would be interesting to study the joint offloading decision-making mechanism of TD and MEC.

7.2. Highly reliable offloading scheduling

TD encounter signal obstruction or even communication interruption during movement, but existing research has not fully considered such situations. The re-established communication connection cause TD to jump to a new MEC service area, which has a greater impact on the dependent partial task offloading paradigm. Therefore, it is necessary to study task offloading scheduling and migration mechanisms with communication failure detection.

7.3. TD offload request activity prediction

Most existing research assumes that requests to offload tasks arrive randomly. However, IoT task offloading usually follows a certain pattern. The prediction of IoT task offload requests should be interesting to study. Thereby improving MEC resource preparation and allocation. Although there are a few studies that predict the data volume of offloading tasks, there is no prediction of the computing resource requirements for task processing. In addition, some studies predict the mobility trend of devices, based on which the nearby EMC nodes can be inferred. It would be an interesting research direction to improve/improve resource preparation and decision-making effects with characteristic task offloading request prediction.

7.4. Practicable partial task offloading

Many studies have considered dividing computing tasks and then offloading part of them to MEC to improve task processing speed and make full use of computing resources. However, such methods lack awareness of differences in task types and unrealistically assume that tasks can be divided into arbitrary sizes. In addition, such research usually estimates CPU resource requirements based on the size of task shards, but the size of the task does not correspond to the computing resources required for the task to be processed. Furthermore, the granularity of task segmentation is also affected by differences in task types. Therefore, it should be study more on partial task offloading optimization with task splitting constraints awareness.

7.5. Multi-objective optimization task offloading with dependencies

When offloading tasks with dependencies, some existing research tends to build serialized models to support processing logic between tasks. However, this is not conducive to parallel processing of tasks on multiple MEC servers. Some studies have considered task-dependent parallel processing, but have not simultaneously considered joint optimization of multiple dimensions such as latency and energy consumption. In addition, some studies do not consider load balancing in Edge-Cloud and support for heterogeneous MEC resources. On the contrary, some dependent task offloading methods that support heterogeneous MEC resources lack optimization of

communication delays. A further study could introduce application type-awareness to improve multi-objective optimization capabilities of constrained task offloading in large-scale heterogeneous computing environments.

7.6. Open MEC architecture

Open-MEC runs open-source software on common hardware platform and uses SDN and virtualization technology to decouple MEC functions from specific physical devices. This way, decoupled MEC functional modules and resources can be reconfigured into customized edge instances. Similarly, the Open Radio Access Network (ORAN) allows MEC to have more control over the RAN to optimize communication resource allocation. Therefore, researchers should conduct more research on resource allocation based on open-source MEC architecture and ORAN to optimize the offloading of IoT computing tasks.

7.7. Aerial access network

Unmanned Aerial Vehicle (UAV) carry wireless communication and computing resources and can serve as MEC computing nodes to provide task offload support for the IoT. In addition, UAVs can also serve as relays for wireless communications to build Air-Ground Collaborative MEC (AGC-MEC) using Reconfigurable Intelligent Ground (RIS) technology. Therefore, it would be very interesting to study MEC computation offloading with air-ground communication support for multi-UAV collaboration.

8. Conclusion

Large numbers of random task offloading requirements and limited MEC resources make the IoT task offloading optimization problem in MEC very complex. Therefore, effective task-offloading and resource allocation mechanisms are needed to maximize task-offloading efficiency and enhance resource utilization. Previous studies have focused on IoT task offloading and MEC resource management respectively. However, there is a lack of comprehensive review on IoT task offloading in MEC. Filling this gap, a comprehensive review is conducted to give a panoramic view of the IoT task offloading mechanism in MEC to help researchers quickly understand the subdivision structure and research paths in this area. The different problems addressed by the task offloading mechanisms and the various approaches proposed to address them are discussed. In addition, the input parameters supported by different offloading methods are explored to analyze the factors affecting the task offloading decision as well as the ability of the offloading mechanism to perceive the realistic environment. Moreover, the problem modeling and the technologies underlying the solutions are carefully examined and categorized. Furthermore, tools and datasets for evaluating task offloading mechanisms and algorithms are also compared and counted. Finally, the open issues and future research directions are presented. According to the research, this is the first time a comprehensive literature review focus on the IoT task offloading in MEC.

Funding statement

The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University, Saudi Arabia for funding this work through large group Research Project under grant number (RGP.2/52/44).

Data availability

No data was used for the research described in the article.

CRediT authorship contribution statement

Wang Dayong: Writing – original draft, Methodology, Conceptualization. **Kamalrulnizam Bin Abu Bakar:** Supervision, Investigation, Conceptualization. **Babangida Isyaku:** Writing – review & editing, Methodology, Conceptualization. **Taiseer Abdalla Elfadil Eisa:** Visualization, Supervision, Methodology, Funding acquisition. **Abdelzahir Abdelmaboud:** Writing – review & editing, Validation, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] B.B. Gupta, M. Quamara, An overview of Internet of Things (IoT): architectural aspects, challenges, and protocols, Concurrency Comput. Pract. Ex. 32 (21) (2020) e4946, https://doi.org/10.1002/cpe.4946.

[2] K.B.A. Bakar, F.T. Zuhra, B. Isyaku, S.B. Sulaiman, A review on the immediate advancement of the internet of things in wireless telecommunications, IEEE Access 11 (2023) 21020–21048, https://doi.org/10.1109/ACCESS.2023.3250466.

[3] C.C. Sobin, A survey on architecture, protocols and challenges in IoT, Wireless Pers. Commun. 112 (3) (2020) 1383–1429, https://doi.org/10.1007/s11277-020-07108-5.

[4] A. Khanna, S. Kaur, Internet of things (IoT), applications and challenges: a comprehensive review, Wireless Pers. Commun. 114 (2) (2020) 1687–1762, https://doi.org/10.1007/s11277-020-07446-4.

[5] A. Yousefpour, G. Ishigaki, R. Gour, J.P. Jue, On reducing IoT service delay via fog offloading, IEEE Internet Things J. 5 (2) (2018) 998–1010, https://doi.org/10.1109/JIOT.2017.2788802.

[6] H. Guo, J. Liu, J. Lv, Toward intelligent task offloading at the edge, IEEE Network 34 (2) (2020) 128–134, https://doi.org/10.1109/MNET.001.1900200.

[7] X. Jin, W. Hua, Z. Wang, Y. Chen, A survey of research on computation offloading in mobile cloud computing, Wireless Network 28 (4) (2022) 1563–1585, https://doi.org/10.1007/s11276-022-02920-2.

[8] C. Wang, R. Guo, H. Yu, Y. Hu, C. Liu, C. Deng, Task offloading in cloud-edge collaboration-based cyber physical machine tool, Robot. Comput.-Integr. Manuf. 79 (2023) 102439, https://doi.org/10.1016/j.rcim.2022.102439.

[9] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, F. Giust, Mobile-edge computing architecture: the role of MEC in the internet of things, IEEE Consum. Electron. Mag. 5 (4) (2016) 84–91, https://doi.org/10.1109/MCE.2016.2590118.

[10] ETSI: V1. 1.1 (2016-03):"Mobile Edge Computing (MEC) – Google Scholar." Accessed: September19, 2023. [Online]. Available: https://scholar.google.com/scholar_lookup?title=GS%20MEC%20001%20-%20V1.1.1%20-%20Mobile%20Edge%20Computing%20(MEC)&author=Etsi&publication_year=2016.

[11] B. Isyaku, K.B.A. Bakar, W. Nagmeldin, A. Abdelmaboud, F. Saeed, F.A. Ghaleb, Reliable failure restoration with Bayesian congestion aware for software defined networks, Accessed: November13, 2023. [Online]. Available: Comput. Syst. Sci. Eng. 46 (3) (2023) https://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=02676192&AN=163012982&h=E8I7V6epghUEg%2BsNuYsm7hKc96OtH9WSCaxlHuEfA1%2FHP9CVbitTLYR%2B2g9BdnFTxQm1iBYgr4Ika8j12cin%2BQ%3D%3D&crl=c.

[12] P. Cruz, N. Achir, A.C. Viana, On the edge of the deployment: a survey on multi-access edge computing, ACM Comput. Surv. 55 (5) (2022) 99:1–99:34, https://doi.org/10.1145/3529758.

[13] M. Songhorabadi, M. Rahimi, A. MoghadamFarid, M. Haghi Kashani, Fog computing approaches in IoT-enabled smart cities, J. Netw. Comput. Appl. 211 (2023) 103557, https://doi.org/10.1016/j.jnca.2022.103557.

[14] A. Ksentini, P.A. Frangoudis, Toward slicing-enabled multi-access edge computing in 5G, IEEE Netw 34 (2) (2020) 99–105, https://doi.org/10.1109/MNET.001.1900261.

[15] B. Jamil, H. Ijaz, M. Shojafar, K. Munir, R. Buyya, Resource allocation and task scheduling in fog computing and internet of everything environments: a taxonomy, review, and future directions, ACM Comput. Surv. 54 (11s) (2022) 233:1–233:38, https://doi.org/10.1145/3513002.

[16] X. Jiang, F.R. Yu, T. Song, V.C.M. Leung, A survey on multi-access edge computing applied to video streaming: some research issues and challenges, IEEE Commun. Surv. Tutor. 23 (2) (2021) 871–903, https://doi.org/10.1109/COMST.2021.3065237.

[17] Q.-V. Pham, A survey of multi-access edge computing in 5G and beyond: fundamentals, technology integration, and state-of-the-art, IEEE Access 8 (2020) 116974–117017, https://doi.org/10.1109/ACCESS.2020.3001277.

[18] J. Ding, M. Nemati, C. Ranaweera, J. Choi, IoT connectivity technologies and applications: a survey, IEEE Access 8 (2020) 67646–67673, https://doi.org/10.1109/ACCESS.2020.2985932.

[19] B. Isyaku, K.B.A. Bakar, Managing smart technologies with software-defined networks for routing and security challenges: a survey, Accessed: November13, 2023. [Online]. Available: Comput. Syst. Sci. Eng. 47 (2) (2023) https://www.researchgate.net/profile/Babangida-Isyaku-3/publication/372669105_Managing_Smart_Technologies_with_Software-Defined_Networks_for_Routing_and_Security_Challenges_A_Survey/links/64c60132213ca521ea183e01/Managing-Smart-Technologies-with-Software-Defined-Networks-for-Routing-and-Security-Challenges-A-Survey.pdf.

[20] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, T. Taleb, Survey on multi-access edge computing for internet of things realization, IEEE Commun. Surv. Tutor. 20 (4) (2018) 2961–2991, https://doi.org/10.1109/COMST.2018.2849509.

[21] A. Hazra, M. Adhikari, T. Amgoth, S.N. Srirama, Joint computation offloading and scheduling optimization of IoT applications in fog networks, IEEE Trans. Netw. Sci. Eng. 7 (4) (2020) 3266–3278, https://doi.org/10.1109/TNSE.2020.3021792.

[22] Q. Peng, C. Wu, Y. Xia, Y. Ma, X. Wang, N. Jiang, DoSRA: a decentralized approach to online edge task scheduling and resource allocation, IEEE Internet Things J. 9 (6) (2022) 4677–4692, https://doi.org/10.1109/JIOT.2021.3107431.

[23] Q. Tang, Distributed task scheduling in serverless edge computing networks for the internet of things: a learning approach, IEEE Internet Things J. 9 (20) (2022) 19634–19648, https://doi.org/10.1109/JIOT.2022.3167417.

[24] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, L. Guo, Computation offloading in mobile edge computing networks: a survey, J. Netw. Comput. Appl. 202 (2022) 103366, https://doi.org/10.1016/j.jnca.2022.103366.

[25] R. Singh, R. Sukapuram, S. Chakraborty, A survey of mobility-aware Multi-access Edge Computing: challenges, use cases and future directions, Ad Hoc Netw. 140 (2023) 103044, https://doi.org/10.1016/j.adhoc.2022.103044.

[26] B. Trinh, G.-M. Muntean, A deep reinforcement learning-based offloading scheme for multi-access edge computing-supported eXtended reality systems, IEEE Trans. Veh. Technol. 72 (1) (2023) 1254–1264, https://doi.org/10.1109/TVT.2022.3207692.

[27] Q. Zhang, M. Lin, L.T. Yang, Z. Chen, S.U. Khan, P. Li, A double deep Q-learning model for energy-efficient edge scheduling, IEEE Trans. Serv. Comput. 12 (5) (2019) 739–749, https://doi.org/10.1109/TSC.2018.2867482.

[28] J. Lim, Latency-aware task scheduling for IoT applications based on artificial intelligence with partitioning in small-scale fog computing environments, Sensors 22 (19) (2022) 19, https://doi.org/10.3390/s22197326.

[29] H. Li, K. Ota, M. Dong, Learning IoT in edge: deep learning for the internet of things with edge computing, IEEE Netw. 32 (1) (2018) 96–101, https://doi.org/10.1109/MNET.2018.1700202.

[30] A. Shakarami, M. Ghobaei-Arani, M. Masdari, M. Hosseinzadeh, A survey on the computation offloading approaches in mobile edge/cloud computing environment: a stochastic-based perspective, J. Grid Comput. 18 (4) (2020) 639–671, https://doi.org/10.1007/s10723-020-09530-2.

[31] K. Xiao, Z. Gao, W. Shi, X. Qiu, Y. Yang, L. Rui, EdgeABC: an architecture for task offloading and resource allocation in the Internet of Things, Future Generat. Comput. Syst. 107 (2020) 498–508, https://doi.org/10.1016/j.future.2020.02.026.

[32] M. Tang, V.W.S. Wong, Deep reinforcement learning for task offloading in mobile edge computing systems, IEEE Trans. Mobile Comput. 21 (6) (2022) 1985–1997, https://doi.org/10.1109/TMC.2020.3036871.

[33] S. Balaji, K. Nathani, R. Santhakumar, IoT technology, applications and challenges: a contemporary survey, Wireless Pers. Commun. 108 (1) (2019) 363–388, https://doi.org/10.1007/s11277-019-06407-w.

[34] G. Javadzadeh, A.M. Rahmani, Fog computing applications in smart cities: a systematic survey, Wireless Network 26 (2) (2020) 1433–1457, https://doi.org/10.1007/s11276-019-02208-y.

[35] B. Brik, K. Dev, Y. Xiao, G. Han, A. Ksentini, Guest editorial introduction to the special section on AI-powered internet of everything (IoE) services in next-generation wireless networks, IEEE Trans. Netw. Sci. Eng 9 (5) (2022) 2952–2954, https://doi.org/10.1109/TNSE.2022.3195385.

[36] J. Iannacci, Internet of things (IoT); internet of everything (IoE); tactile internet; 5G – a (not so evanescent) unifying vision empowered by EH-MEMS (energy harvesting MEMS) and RF-MEMS (radio frequency MEMS), Sens. Actuators Phys. 272 (2018) 187–198, https://doi.org/10.1016/j.sna.2018.01.038.

[37] C. Li, Q. Zhang, C. Huang, Y. Luo, Optimal service selection and placement based on popularity and server load in multi-access edge computing, J. Netw. Syst. Manag. 31 (1) (2022) 15, https://doi.org/10.1007/s10922-022-09703-2.

[38] Q. Duan, S. Wang, N. Ansari, Convergence of networking and cloud/edge computing: status, challenges, and opportunities, IEEE Netw 34 (6) (2020) 148–155, https://doi.org/10.1109/MNET.011.2000089.

[39] M. Ahmed, et al., A survey on vehicular task offloading: classification, issues, and challenges, J. King Saud Univ. - Comput. Inf. Sci. 34 (2022), https://doi.org/10.1016/j.jksuci.2022.05.016.

[40] Y. Miao, G. Wu, M. Li, A. Ghoneim, M. Al-Rakhami, M.S. Hossain, Intelligent task prediction and computation offloading based on mobile-edge cloud computing, Future Generat. Comput. Syst. 102 (2020) 925–931, https://doi.org/10.1016/j.future.2019.09.035.

[41] S. Douch, M.R. Abid, K. Zine-Dine, D. Bouzidi, D. Benhaddou, Edge computing technology enablers: a systematic lecture study, IEEE Access 10 (2022) 69264–69302, https://doi.org/10.1109/ACCESS.2022.3183634.

[42] G. Kaur, R.S. Batth, Edge computing: classification, applications, and challenges, in: 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM), 2021, pp. 254–259, https://doi.org/10.1109/ICIEM51511.2021.9445331.

[43] G. Russo Russo, V. Cardellini, F. Lo Presti, A framework for offloading and migration of serverless functions in the Edge–Cloud Continuum, Pervasive Mob. Comput. 100 (2024) 101915, https://doi.org/10.1016/j.pmcj.2024.101915.

[44] G. Russo Russo, D. Ferrarelli, D. Pasquali, V. Cardellini, F. Lo Presti, QoS-aware offloading policies for serverless functions in the Cloud-to-Edge continuum, Future Generat. Comput. Syst. 156 (2024) 1–15, https://doi.org/10.1016/j.future.2024.02.019.

[45] G. Qiao, S. Leng, Y. Zhang, Online learning and optimization for computation offloading in D2D edge computing and networks, Mobile Network. Appl. 27 (3) (2022) 1111–1122, https://doi.org/10.1007/s11036-018-1176-y.

[46] J. Xie, Y. Jia, W. Wen, Z. Chen, L. Liang, Dynamic D2D multihop offloading in multi-access edge computing from the perspective of learning theory in games, IEEE Trans. Netw. Serv. Manag. 20 (1) (2023) 305–318, https://doi.org/10.1109/TNSM.2022.3201470.

[47] M.H. Adnan, Z.A. Zukarnain, O.A. Amodu, Fundamental design aspects of UAV-enabled MEC systems: a review on models, challenges, and future opportunities, Comput. Sci. Rev. 51 (2024) 100615, https://doi.org/10.1016/j.cosrev.2023.100615.

[48] Y. Zhang, H. Zhang, K. Sun, J. Huo, N. Wang, V.C.M. Leung, Partial computation offloading in satellite based three-tier cloud-edge integration networks, IEEE Trans. Wireless Commun. (2023) 1, https://doi.org/10.1109/TWC.2023.3282630, 1.

[49] L. Zhao, A digital twin-assisted intelligent partial offloading approach for vehicular edge computing, IEEE J. Sel. Area. Commun. 41 (11) (2023) 3386–3400, https://doi.org/10.1109/JSAC.2023.3310062.

[50] H. Lin, S. Zeadally, Z. Chen, H. Labiod, L. Wang, A survey on computation offloading modeling for edge computing, J. Netw. Comput. Appl. 169 (2020) 102781, https://doi.org/10.1016/j.jnca.2020.102781.

[51] H. Wu, Multi-objective decision-making for mobile cloud offloading: a survey, IEEE Access 6 (2018) 3962–3976, https://doi.org/10.1109/ACCESS.2018.2791504.

[52] Z. He, Y. Xu, D. Liu, W. Zhou, K. Li, Energy-efficient computation offloading strategy with task priority in cloud assisted multi-access edge computing, Future Generat. Comput. Syst. 148 (2023) 298–313, https://doi.org/10.1016/j.future.2023.06.014.

[53] C. Li, J. Tang, Y. Zhang, X. Yan, Y. Luo, Energy efficient computation offloading for nonorthogonal multiple access assisted mobile edge computing with energy harvesting devices, Comput. Network. 164 (2019) 106890, https://doi.org/10.1016/j.comnet.2019.106890.

[54] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, P. Mohapatra, Edge cloud offloading algorithms: issues, methods, and perspectives, ACM Comput. Surv. 52 (1) (2019) 2:1–2:23, https://doi.org/10.1145/3284387.

[55] Z. Sun, Y. Mo, C. Yu, Graph-reinforcement-learning-based task offloading for multiaccess edge computing, IEEE Internet Things J. 10 (4) (2023) 3138–3150, https://doi.org/10.1109/JIOT.2021.3123822.

[56] R. Singh, S. Armour, A. Khan, M. Sooriyabandara, G. Oikonomou, Heuristic approaches for computational offloading in multi-access edge computing networks, in: 2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications, 2020, pp. 1–7, https://doi.org/10.1109/PIMRC48278.2020.9217181.

[57] C.-Y. Hsieh, Y. Ren, J.-C. Chen, Edge-cloud offloading: knapsack potential game in 5G multi-access edge computing, IEEE Trans. Wireless Commun. (2023) 1, https://doi.org/10.1109/TWC.2023.3248270, 1.

[58] M.N. Yusuf, K. Bin Abu Bakar, B. Isyaku, F. Mukhlif, Distributed controller placement in software-defined networks with consistency and interoperability problems, J. Electr. Comput. Eng. 2023 (2023). Accessed: November8, 2023. [Online]. Available: https://www.hindawi.com/journals/jece/2023/6466996/.

[59] H. Lin, X. Xu, J. Zhao, X. Wang, Dynamic service migration in ultra-dense multi-access edge computing network for high-mobility scenarios, EURASIP J. Wirel. Commun. Netw. 2020 (1) (2020) 191, https://doi.org/10.1186/s13638-020-01805-2.

[60] L.A. Haibeh, M.C.E. Yagoub, A. Jarray, A survey on mobile edge computing infrastructure: design, resource management, and optimization approaches, IEEE Access 10 (2022) 27591–27610, https://doi.org/10.1109/ACCESS.2022.3152787.

[61] B. Isyaku, K.B.A. Bakar, F.A. Ghaleb, A. Al-Nahari, Dynamic routing and failure recovery approaches for efficient resource utilization in OpenFlow-SDN: a survey, IEEE Access 10 (2022) 121791–121815, https://doi.org/10.1109/ACCESS.2022.3222849.

[62] J. Ren, D. Zhang, S. He, Y. Zhang, T. Li, A survey on end-edge-cloud orchestrated network computing paradigms: transparent computing, mobile edge computing, fog computing, and cloudlet, ACM Comput. Surv. 52 (6) (2019) 125:1–125:36, https://doi.org/10.1145/3362031.

[63] J.A. Hurtado Sánchez, K. Casilimas, O.M. Caicedo Rendon, Deep reinforcement learning for resource management on network slicing: a survey, Sensors 22 (8) (2022) 8, https://doi.org/10.3390/s22083031.

[64] M. Adhikari, A. Munusamy, N. Kumar, S.N. Srirama, Cybertwin-driven resource provisioning for IoE applications at 6G-enabled edge networks, IEEE Trans. Ind. Inf. 18 (7) (2022) 4850–4858, https://doi.org/10.1109/TII.2021.3096672.

[65] H. Tabatabaee Malazi, et al., Dynamic service placement in multi-access edge computing: a systematic literature review, IEEE Access 10 (2022) 32639–32688, https://doi.org/10.1109/ACCESS.2022.3160738.

[66] J. von Mankowski, E. Durmaz, A. Papa, H. Vijayaraghavan, W. Kellerer, Aerial-aided multiaccess edge computing: dynamic and joint optimization of task and service placement and routing in multilayer networks, IEEE Trans. Aero. Electron. Syst. 59 (3) (2023) 2593–2607, https://doi.org/10.1109/TAES.2022.3217430.

[67] M. Maray, J. Shuja, Computation offloading in mobile cloud computing and mobile edge computing: survey, taxonomy, and open issues, Mobile Inf. Syst. 2022 (2022) e1121822, https://doi.org/10.1155/2022/1121822.

[68] B. Cao, L. Zhang, Y. Li, D. Feng, W. Cao, Intelligent offloading in multi-access edge computing: a state-of-the-art review and framework, IEEE Commun. Mag. 57 (3) (2019) 56–62, https://doi.org/10.1109/MCOM.2019.1800608.

[69] W. Sun, H. Zhang, R. Wang, Y. Zhang, Reducing offloading latency for digital twin edge networks in 6G, IEEE Trans. Veh. Technol. 69 (10) (2020) 12240–12251, https://doi.org/10.1109/TVT.2020.3018817.

[70] M.D. Hossain, et al., Fuzzy decision-based efficient task offloading management scheme in multi-tier MEC-enabled networks, Sensors 21 (4) (2021) 4, https://doi.org/10.3390/s21041484.

[71] A. Shakarami, M. Ghobaei-Arani, A. Shahidinejad, A survey on the computation offloading approaches in mobile edge computing: a machine learning-based perspective, Comput. Network. 182 (2020) 107496, https://doi.org/10.1016/j.comnet.2020.107496.

[72] X. Chen, Z. Li, Y. Chen, X. Wang, Performance analysis and uplink scheduling for QoS-aware NB-IoT networks in mobile computing, IEEE Access 7 (2019) 44404–44415, https://doi.org/10.1109/ACCESS.2019.2908985.

[73] A. Heidari, M.A.J. Jamali, N.J. Navimipour, S. Akbarpour, A QoS-aware technique for computation offloading in IoT-edge platforms using a convolutional neural network and Markov decision process, IT Prof. 25 (1) (2023) 24–39, https://doi.org/10.1109/MITP.2022.3217886.

[74] K. Peng, H. Huang, B. Zhao, A. Jolfaei, X. Xu, M. Bilal, Intelligent computation offloading and resource allocation in IIoT with end-edge-cloud computing using NSGA-III, IEEE Trans. Netw. Sci. Eng. (2022) 1, https://doi.org/10.1109/TNSE.2022.3155490, 1.

[75] S. Aljanabi, A. Chalechale, Improving IoT services using a hybrid fog-cloud offloading, IEEE Access 9 (2021) 13775–13788, https://doi.org/10.1109/ACCESS.2021.3052458.

[76] H. Jin, M.A. Gregory, S. Li, A review of intelligent computation offloading in multiaccess edge computing, IEEE Access 10 (2022) 71481–71495, https://doi.org/10.1109/ACCESS.2022.3187701.

[77] M.Y. Akhlaqi, Z.B. Mohd Hanapi, Task offloading paradigm in mobile edge computing-current issues, adopted approaches, and future directions, J. Netw. Comput. Appl. 212 (2023) 103568, https://doi.org/10.1016/j.jnca.2022.103568.

[78] H. Chen, W. Qin, L. Wang, Task partitioning and offloading in IoT cloud-edge collaborative computing framework: a survey, J. Cloud Comput. 11 (1) (2022) 86, https://doi.org/10.1186/s13677-022-00365-8.

[79] A. Heidari, M.A. Jabraeil Jamali, N. Jafari Navimipour, S. Akbarpour, Internet of Things offloading: ongoing issues, opportunities, and future challenges, Int. J. Commun. Syst. 33 (14) (2020) e4474, https://doi.org/10.1002/dac.4474.

[80] M. Aazam, S. Zeadally, K.A. Harras, Offloading in fog computing for IoT: review, enabling technologies, and research opportunities, Future Generat. Comput. Syst. 87 (2018) 278–289, https://doi.org/10.1016/j.future.2018.04.057.

[81] K. Lone, S.A. Sofi, "A review on offloading in fog-based Internet of Things: architecture, machine learning approaches, and open issues," High-Confid, Comput. Times 3 (2) (2023) 100124, https://doi.org/10.1016/j.hcc.2023.100124.

[82] A. Islam, A. Debnath, M. Ghose, S. Chakraborty, A survey on task offloading in multi-access edge computing, J. Syst. Architect. 118 (2021) 102225, https://doi.org/10.1016/j.sysarc.2021.102225.

[83] P.X. Nguyen, Backscatter-assisted data offloading in OFDMA-based wireless-powered mobile edge computing for IoT networks, IEEE Internet Things J. 8 (11) (2021) 9233–9243, https://doi.org/10.1109/JIOT.2021.3057360.

[84] A. Samanta, J. Tang, Dyme: dynamic microservice scheduling in edge computing enabled IoT, IEEE Internet Things J. 7 (7) (2020) 6164–6174, https://doi.org/10.1109/JIOT.2020.2981958.

[85] Y. Cheng, H. Zhao, W. Xia, Energy-aware offloading and power optimization in full-duplex mobile edge computing-enabled cellular IoT networks, IEEE Sens. J. 22 (24) (2022) 24607–24618, https://doi.org/10.1109/JSEN.2022.3218584.

[86] Z. Yu, Y. Gong, S. Gong, Y. Guo, Joint task offloading and resource allocation in UAV-enabled mobile edge computing, IEEE Internet Things J. 7 (4) (2020) 3147–3159, https://doi.org/10.1109/JIOT.2020.2965898.

[87] Z. Yao, H. Wu, Y. Chen, Multi-objective cooperative computation offloading for MEC in UAVs hybrid networks via integrated optimization framework, Comput. Commun. 202 (2023) 124–134, https://doi.org/10.1016/j.comcom.2023.01.006.

[88] B. Liu, C. Liu, M. Peng, Resource allocation for energy-efficient MEC in NOMA-enabled massive IoT networks, IEEE J. Sel. Area. Commun. 39 (4) (2021) 1015–1027, https://doi.org/10.1109/JSAC.2020.3018809.

[89] S. Liu, Satisfaction-maximized secure computation offloading in multi-eavesdropper MEC networks, IEEE Trans. Wireless Commun. 21 (6) (2022) 4227–4241, https://doi.org/10.1109/TWC.2021.3128247.

[90] S. Azizi, M. Othman, H. Khamfroush, DECO: a deadline-aware and energy-efficient algorithm for task offloading in mobile edge computing, IEEE Syst. J. 17 (1) (2023) 952–963, https://doi.org/10.1109/JSYST.2022.3185011.

[91] C.-W. Hsu, Y.-L. Hsu, H.-Y. Wei, Energy-efficient edge offloading in heterogeneous industrial IoT networks for factory of future, IEEE Access 8 (2020) 183035–183050, https://doi.org/10.1109/ACCESS.2020.3029253.

[92] J. Li, Maximizing user service satisfaction for delay-sensitive IoT applications in edge computing, IEEE Trans. Parallel Distr. Syst. 33 (5) (2022) 1199–1212, https://doi.org/10.1109/TPDS.2021.3107137.

[93] T.D.T. Nguyen, V. Nguyen, V.-N. Pham, L.N.T. Huynh, Md D. Hossain, E.-N. Huh, Modeling data redundancy and cost-aware task allocation in MEC-enabled internet-of-vehicles applications, IEEE Internet Things J. 8 (3) (2021) 1687–1701, https://doi.org/10.1109/JIOT.2020.3015534.

[94] C. Tang, C. Zhu, N. Zhang, M. Guizani, J.J.P.C. Rodrigues, SDN-assisted mobile edge computing for collaborative computation offloading in industrial internet of things, IEEE Internet Things J. 9 (23) (2022) 24253–24263, https://doi.org/10.1109/JIOT.2022.3190281.

[95] F. Hoseiny, S. Azizi, M. Shojafar, R. Tafazolli, Joint QoS-aware and cost-efficient task scheduling for fog-cloud resources in a volunteer computing system, ACM Trans. Internet Technol. 21 (4) (2021) 86:1–86:21, https://doi.org/10.1145/3418501.

[96] C. Chen, Y. Zeng, H. Li, Y. Liu, S. Wan, A multihop task offloading decision model in MEC-enabled internet of vehicles, IEEE Internet Things J. 10 (4) (2023) 3215–3230, https://doi.org/10.1109/JIOT.2022.3143529.

[97] J. Liu, Y. Zhang, J. Ren, Y. Zhang, Auction-based dependent task offloading for IoT users in edge clouds, IEEE Internet Things J. 10 (6) (2023) 4907–4921, https://doi.org/10.1109/JIOT.2022.3221431.

[98] A. Samanta, F. Esposito, T.G. Nguyen, Fault-tolerant mechanism for edge-based IoT networks with demand uncertainty, IEEE Internet Things J. 8 (23) (2021) 16963–16971, https://doi.org/10.1109/JIOT.2021.3075681.

[99] Z. Jia, Q. Wu, C. Dong, C. Yuen, Z. Han, Hierarchical aerial computing for internet of things via cooperation of HAPs and UAVs, IEEE Internet Things J. 10 (7) (2023) 5676–5688, https://doi.org/10.1109/JIOT.2022.3151639.

[100] Y. Hao, Z. Song, Z. Zheng, Q. Zhang, Z. Miao, Joint communication, computing, and caching resource allocation in LEO satellite MEC networks, IEEE Access 11 (2023) 6708–6716, https://doi.org/10.1109/ACCESS.2023.3237701.

[101] A. Gao, Q. Wang, Y. Hu, W. Duan, An offloading optimization scheme for multi-UAV aided network in mobile computing, in: 2020 International Wireless Communications and Mobile Computing (IWCMC), 2020, pp. 1468–1473, https://doi.org/10.1109/IWCMC48107.2020.9148136.

[102] L. Lei, H. Xu, X. Xiong, K. Zheng, W. Xiang, Joint computation offloading and multiuser scheduling using approximate dynamic programming in NB-IoT edge computing system, IEEE Internet Things J. 6 (3) (2019) 5345–5362, https://doi.org/10.1109/JIOT.2019.2900550.

[103] X. An, R. Fan, H. Hu, N. Zhang, S. Atapattu, T.A. Tsiftsis, Joint task offloading and resource allocation for IoT edge computing with sequential task dependency, IEEE Internet Things J. 9 (17) (2022) 16546–16561, https://doi.org/10.1109/JIOT.2022.3150976.

[104] H. Wang, Low-complexity and efficient dependent subtask offloading strategy in IoT integrated with multi-access edge computing, IEEE Trans. Netw. Serv. Manag. (2023) 1, https://doi.org/10.1109/TNSM.2023.3295653, 1.

[105] Q. Wang, A. Gao, Y. Hu, Joint power and QoE optimization scheme for multi-UAV assisted offloading in mobile computing, IEEE Access 9 (2021) 21206–21217, https://doi.org/10.1109/ACCESS.2021.3055335.

[106] Y. Wang, J. Yang, X. Guo, Z. Qu, A game-theoretic approach to computation offloading in satellite edge computing, IEEE Access 8 (2020) 12510–12520, https://doi.org/10.1109/ACCESS.2019.2963068.

[107] X. Yang, H. Luo, Y. Sun, J. Zou, M. Guizani, Coalitional game-based cooperative computation offloading in MEC for reusable tasks, IEEE Internet Things J. 8 (16) (2021) 12968–12982, https://doi.org/10.1109/JIOT.2021.3064186.

[108] S. Hu, G. Li, Dynamic request scheduling optimization in mobile edge computing for IoT applications, IEEE Internet Things J. 7 (2) (2020) 1426–1437, https://doi.org/10.1109/JIOT.2019.2955311.

[109] X. Yuan, H. Tian, H. Wang, H. Su, J. Liu, A. Taherkordi, Edge-enabled WBANs for efficient QoS provisioning healthcare monitoring: a two-stage potential game-based computation offloading strategy, IEEE Access 8 (2020) 92718–92730, https://doi.org/10.1109/ACCESS.2020.2992639.

[110] W. Fan, L. Yao, J. Han, F. Wu, Y. Liu, Game-based multitype task offloading among mobile-edge-computing-enabled base stations, IEEE Internet Things J. 8 (24) (2021) 17691–17704, https://doi.org/10.1109/JIOT.2021.3082291.

[111] F. Algarni, A novel quality-based computation offloading framework for edge cloud-supported internet of things, Alex. Eng. J. 70 (2023) 585–599, https://doi.org/10.1016/j.aej.2023.03.026.

[112] Z. Ai, W. Zhang, M. Li, P. Li, L. Shi, A smart collaborative framework for dynamic multi-task offloading in IIoT-MEC networks, Peer–Peer Netw. Appl. 16 (2) (2023) 749–764, https://doi.org/10.1007/s12083-022-01441-1.

[113] A. Acheampong, Y. Zhang, X. Xu, A parallel computing based model for online binary computation offloading in mobile edge computing, Comput. Commun. 203 (2023) 248–261, https://doi.org/10.1016/j.comcom.2023.03.004.

[114] H. Wu, Z. Zhang, C. Guan, K. Wolter, M. Xu, Collaborate edge and cloud computing with distributed deep learning for smart city internet of things, IEEE Internet Things J. 7 (9) (2020) 8099–8110, https://doi.org/10.1109/JIOT.2020.2996784.

[115] Z. Wang, T. Lv, Z. Chang, Computation offloading and resource allocation based on distributed deep learning and software defined mobile edge computing, Comput. Network. 205 (2022) 108732, https://doi.org/10.1016/j.comnet.2021.108732.

[116] J. Niu, S. Zhang, K. Chi, G. Shen, W. Gao, Deep learning for online computation offloading and resource allocation in NOMA, Comput. Network. 216 (2022) 109238, https://doi.org/10.1016/j.comnet.2022.109238.

[117] Z. Ali, Z.H. Abbas, G. Abbas, A. Numani, M. Bilal, Smart computational offloading for mobile edge computing in next-generation Internet of Things networks, Comput. Network. 198 (2021) 108356, https://doi.org/10.1016/j.comnet.2021.108356.

[118] T.T. Khanh, T.H. Hai, M.D. Hossain, E.-N. Huh, Fuzzy-assisted mobile edge orchestrator and SARSA learning for flexible offloading in heterogeneous IoT environment, Sensors 22 (13) (2022) 13, https://doi.org/10.3390/s22134727.

[119] Y. Cui, D. Zhang, T. Zhang, L. Chen, M. Piao, H. Zhu, Novel method of mobile edge computation offloading based on evolutionary game strategy for IoT devices, AEU – Int. J. Electron. Commun. 118 (2020) 153134, https://doi.org/10.1016/j.aeue.2020.153134.
[120] M. Yi, P. Yang, M. Chen, N.T. Loc, A DRL-driven intelligent joint optimization strategy for computation offloading and resource allocation in ubiquitous edge IoT systems, IEEE Trans. Emerg. Top. Comput. Intell. 7 (1) (2023) 39–54, https://doi.org/10.1109/TETCI.2022.3193367.
[121] A. Heidari, N.J. Navimipour, M.A.J. Jamali, S. Akbarpour, A green, secure, and deep intelligent method for dynamic IoT-edge-cloud offloading scenarios, Sustain. Comput. Inf. Syst. 38 (2023) 100859, https://doi.org/10.1016/j.suscom.2023.100859.
[122] A. Heidari, N.J. Navimipour, M.A.J. Jamali, S. Akbarpour, A hybrid approach for latency and battery lifetime optimization in IoT devices through offloading and CNN learning, Sustain. Comput. Inf. Syst. 39 (2023) 100899, https://doi.org/10.1016/j.suscom.2023.100899.
[123] I. Khan, X. Tao, G.M.S. Rahman, W.U. Rehman, T. Salam, Advanced energy-efficient computation offloading using deep reinforcement learning in MTC edge computing, IEEE Access 8 (2020) 82867–82875, https://doi.org/10.1109/ACCESS.2020.2991057.
[124] M. Li, Cloud–edge collaborative resource allocation for blockchain-enabled internet of things: a collective reinforcement learning approach, IEEE Internet Things J. 9 (22) (2022) 23115–23129, https://doi.org/10.1109/JIOT.2022.3185289.
[125] Y. Liu, H. Yu, S. Xie, Y. Zhang, Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks, IEEE Trans. Veh. Technol. 68 (11) (2019) 11158–11168, https://doi.org/10.1109/TVT.2019.2935450.
[126] L. Huang, S. Bi, Y.-J.A. Zhang, Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks, IEEE Trans. Mobile Comput. 19 (11) (2020) 2581–2593, https://doi.org/10.1109/TMC.2019.2928811.
[127] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, L. Li, Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning, IEEE Trans. Cogn. Commun. Netw. 7 (3) (2021) 881–892, https://doi.org/10.1109/TCCN.2021.3066619.
[128] S. Yu, X. Chen, L. Yang, D. Wu, M. Bennis, J. Zhang, Intelligent edge: leveraging deep imitation learning for mobile edge computation offloading, IEEE Wireless Commun. 27 (1) (2020) 92–99, https://doi.org/10.1109/MWC.001.1900232.
[129] L. Lei, H. Xu, X. Xiong, K. Zheng, W. Xiang, X. Wang, Multiuser resource control with deep reinforcement learning in IoT edge computing, IEEE Internet Things J. 6 (6) (2019) 10119–10133, https://doi.org/10.1109/JIOT.2019.2935543.
[130] L. Qian, Y. Wu, F. Jiang, N. Yu, W. Lu, B. Lin, NOMA assisted multi-task multi-access mobile edge computing via deep reinforcement learning for industrial internet of things, IEEE Trans. Ind. Inf. 17 (8) (2021) 5688–5698, https://doi.org/10.1109/TII.2020.3001355.
[131] D.C. Nguyen, P.N. Pathirana, M. Ding, A. Seneviratne, Privacy-preserved task offloading in mobile blockchain with deep reinforcement learning, IEEE Trans. Netw. Serv. Manag. 17 (4) (2020) 2536–2549, https://doi.org/10.1109/TNSM.2020.3010967.
[132] S. Wan, J. Lu, P. Fan, K.B. Letaief, Toward big data processing in IoT: path planning and resource management of UAV base stations in mobile-edge computing system, IEEE Internet Things J. 7 (7) (2020) 5995–6009, https://doi.org/10.1109/JIOT.2019.2954825.
[133] D. Wei, N. Xi, J. Ma, L. He, UAV-assisted privacy-preserving online computation offloading for internet of things, Rem. Sens. 13 (23) (2021) 23, https://doi.org/10.3390/rs13234853.
[134] J. Xu, B. Ai, L. Chen, Y. Cui, N. Wang, Deep reinforcement learning for computation and communication resource allocation in multiaccess MEC assisted Railway IoT networks, IEEE Trans. Intell. Transport. Syst. 23 (12) (2022) 23797–23808, https://doi.org/10.1109/TITS.2022.3205175.
[135] D.C. Nguyen, P.N. Pathirana, M. Ding, A. Seneviratne, Secure computation offloading in blockchain based IoT networks with deep reinforcement learning, IEEE Trans. Netw. Sci. Eng. 8 (4) (2021) 3192–3208, https://doi.org/10.1109/TNSE.2021.3106956.
[136] S. Yang, L. Zhang, L. Cui, Q. Dong, W. Xiao, C. Luo, RLCS: towards a robust and efficient mobile edge computing resource scheduling and task offloading system based on graph neural network, Comput. Commun. 206 (2023) 38–50, https://doi.org/10.1016/j.comcom.2023.04.020.
[137] M.A. Ebrahim, G.A. Ebrahim, H.K. Mohamed, S.O. Abdellatif, A deep learning approach for task offloading in multi-UAV aided mobile edge computing, IEEE Access 10 (2022) 101716–101731, https://doi.org/10.1109/ACCESS.2022.3208584.
[138] Y. Wu, Secrecy-based delay-aware computation offloading via mobile edge computing for internet of things, IEEE Internet Things J. 6 (3) (2019) 4201–4213, https://doi.org/10.1109/JIOT.2018.2875241.
[139] W.-Z. Zhang, Secure and optimized load balancing for multitier IoT and edge-cloud computing systems, IEEE Internet Things J. 8 (10) (2021) 8119–8132, https://doi.org/10.1109/JIOT.2020.3042433.
[140] L. Ma, X. Wang, X. Wang, L. Wang, Y. Shi, M. Huang, TCDA: truthful combinatorial double auctions for mobile edge computing in industrial internet of things, IEEE Trans. Mobile Comput. 21 (11) (2022) 4125–4138, https://doi.org/10.1109/TMC.2021.3064314.
[141] J. Liu, Q. Zhang, Computation resource allocation for heterogeneous time-critical IoT services in MEC, in: 2020 IEEE Wireless Communications and Networking Conference (WCNC), 2020, pp. 1–6, https://doi.org/10.1109/WCNC45663.2020.9120832.
[142] Z. Tong, J. Cai, J. Mei, K. Li, K. Li, Dynamic energy-saving offloading strategy guided by Lyapunov optimization for IoT devices, IEEE Internet Things J. 9 (20) (2022) 19903–19915, https://doi.org/10.1109/JIOT.2022.3168968.
[143] H. Wu, K. Wolter, P. Jiao, Y. Deng, Y. Zhao, M. Xu, EEDTO: an energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing, IEEE Internet Things J. 8 (4) (2021) 2163–2176, https://doi.org/10.1109/JIOT.2020.3033521.
[144] J. Chen, H. Wu, R. Li, P. Jiao, Green parallel online offloading for DSCI-type tasks in IoT-edge systems, IEEE Trans. Ind. Inf. 18 (11) (2022) 7955–7966, https://doi.org/10.1109/TII.2022.3167668.
[145] L. Liu, X. Yuan, D. Chen, N. Zhang, H. Sun, A. Taherkordi, Multi-user dynamic computation offloading and resource allocation in 5G MEC heterogeneous networks with static and dynamic subchannels, IEEE Trans. Veh. Technol. (2023) 1–16, https://doi.org/10.1109/TVT.2023.3285069.
[146] X. Lyu, Optimal schedule of mobile edge computing for internet of things using partial information, IEEE J. Sel. Area. Commun. 35 (11) (2017) 2606–2615, https://doi.org/10.1109/JSAC.2017.2760186.
[147] Y. Deng, Z. Chen, X. Yao, S. Hassan, Ali M.A. Ibrahim, Parallel offloading in green and sustainable mobile edge computing for delay-constrained IoT system, IEEE Trans. Veh. Technol. 68 (12) (2019) 12202–12214, https://doi.org/10.1109/TVT.2019.2944926.
[148] A.M. Seid, J. Lu, H.N. Abishu, T.A. Ayall, Blockchain-enabled task offloading with energy harvesting in multi-UAV-assisted IoT networks: a multi-agent DRL approach, IEEE J. Sel. Area. Commun. 40 (12) (2022) 3517–3532, https://doi.org/10.1109/JSAC.2022.3213352.
[149] H. Mai Do, T.P. Tran, M. Yoo, Deep reinforcement learning-based task offloading and resource allocation for industrial IoT in MEC federation system, IEEE Access 11 (2023) 83150–83170, https://doi.org/10.1109/ACCESS.2023.3302518.
[150] H. Ke, J. Wang, H. Wang, Y. Ge, Joint optimization of data offloading and resource allocation with renewable energy aware for IoT devices: a deep reinforcement learning approach, IEEE Access 7 (2019) 179349–179363, https://doi.org/10.1109/ACCESS.2019.2959348.
[151] A.M. Seid, G.O. Boateng, B. Mareri, G. Sun, W. Jiang, Multi-agent DRL for task offloading and resource allocation in multi-UAV enabled IoT edge network, IEEE Trans. Netw. Serv. Manag. 18 (4) (2021) 4531–4547, https://doi.org/10.1109/TNSM.2021.3096673.
[152] D.S. Lakew, A.-T. Tran, N.-N. Dao, S. Cho, Intelligent offloading and resource allocation in heterogeneous aerial access IoT networks, IEEE Internet Things J. 10 (7) (2023) 5704–5718, https://doi.org/10.1109/JIOT.2022.3161571.
[153] G. Wu, Z. Xu, H. Zhang, S. Shen, S. Yu, Multi-agent DRL for joint completion delay and energy consumption with queuing theory in MEC-based IIoT, J. Parallel Distr. Comput. 176 (2023) 80–94, https://doi.org/10.1016/j.jpdc.2023.02.008.
[154] X. Zhu, Y. Luo, A. Liu, M.Z.A. Bhuiyan, S. Zhang, Multiagent deep reinforcement learning for vehicular computation offloading in IoT, IEEE Internet Things J. 8 (12) (2021) 9763–9773, https://doi.org/10.1109/JIOT.2020.3040768.
[155] Z. Li, M. Xu, J. Nie, J. Kang, W. Chen, S. Xie, NOMA-enabled cooperative computation offloading for blockchain-empowered internet of things: a learning approach, IEEE Internet Things J. 8 (4) (2021) 2364–2378, https://doi.org/10.1109/JIOT.2020.3016644.
[156] N. Cheng, Space/aerial-assisted computing offloading for IoT applications: a learning-based approach, IEEE J. Sel. Area. Commun. 37 (5) (2019) 1117–1129, https://doi.org/10.1109/JSAC.2019.2906789.
[157] S. Chen, L. Rui, Z. Gao, W. Li, X. Qiu, Cache-assisted collaborative task offloading and resource allocation strategy: a metareinforcement learning approach, IEEE Internet Things J. 9 (20) (2022) 19823–19842, https://doi.org/10.1109/JIOT.2022.3168885.

[158] L. Zang, X. Zhang, B. Guo, Federated deep reinforcement learning for online task offloading and resource allocation in WPC-mec networks, IEEE Access 10 (2022) 9856–9867, https://doi.org/10.1109/ACCESS.2022.3144415.
[159] X. Chen, G. Liu, Federated deep reinforcement learning-based task offloading and resource allocation for smart cities in a mobile edge network, Sensors 22 (13) (2022) 13, https://doi.org/10.3390/s22134738.
[160] S. Wu, H. Xue, L. Zhang, Q-Learning-Aided offloading strategy in edge-assisted federated learning over industrial IoT, Electronics 12 (7) (2023) 7, https://doi.org/10.3390/electronics12071706.
[161] H. Seo, H. Oh, J.K. Choi, S. Park, Differential pricing-based task offloading for delay-sensitive IoT applications in mobile edge computing system, IEEE Internet Things J. 9 (19) (2022) 19116–19131, https://doi.org/10.1109/JIOT.2022.3163820.
[162] H. Guo, J. Liu, J. Zhang, Computation offloading for multi-access mobile edge computing in ultra-dense networks, IEEE Commun. Mag. 56 (8) (2018) 14–19, https://doi.org/10.1109/MCOM.2018.1701069.
[163] S.K.U. Zaman, et al., COME-UP: computation offloading in mobile edge computing with LSTM based user direction prediction, Appl. Sci. 12 (7) (2022) 3312, https://doi.org/10.3390/app12073312.
[164] X. Deng, Z. Sun, D. Li, J. Luo, S. Wan, User-centric computation offloading for edge computing, IEEE Internet Things J. 8 (16) (2021) 12559–12568, https://doi.org/10.1109/JIOT.2021.3057694.
[165] V.S. Pana, O.P. Babalola, V. Balyan, 5G radio access networks: a survey, Array 14 (2022) 100170, https://doi.org/10.1016/j.array.2022.100170.
[166] H. Hu, W. Song, Q. Wang, R.Q. Hu, H. Zhu, Energy efficiency and delay tradeoff in an MEC-enabled mobile IoT network, IEEE Internet Things J. 9 (17) (2022) 15942–15956, https://doi.org/10.1109/JIOT.2022.3153847.
[167] Y. Yang, Y. Gong, Y.-C. Wu, Intelligent-reflecting-surface-aided mobile edge computing with binary offloading: energy minimization for IoT devices, IEEE Internet Things J. 9 (15) (2022) 12973–12983, https://doi.org/10.1109/JIOT.2022.3173027.
[168] Y. Gao, W. Tang, M. Wu, P. Yang, L. Dan, Dynamic social-aware computation offloading for low-latency communications in IoT, IEEE Internet Things J. 6 (5) (2019) 7864–7877, https://doi.org/10.1109/JIOT.2019.2909299.
[169] S. Mendonça, B. Damásio, L. Charlita de Freitas, L. Oliveira, M. Cichy, A. Nicita, The rise of 5G technologies and systems: a quantitative analysis of knowledge production, Telecommun. Pol. 46 (4) (2022) 102327, https://doi.org/10.1016/j.telpol.2022.102327.
[170] F. Zhou, L. Feng, M. Kadoch, P. Yu, W. Li, Z. Wang, Multiagent RL aided task offloading and resource management in Wi-fi 6 and 5G coexisting industrial wireless environment, IEEE Trans. Ind. Inf. 18 (5) (2022) 2923–2933, https://doi.org/10.1109/TII.2021.3106973.
[171] X. Xu, D. Li, Z. Dai, S. Li, X. Chen, A heuristic offloading method for deep learning edge services in 5G networks, IEEE Access 7 (2019) 67734–67744, https://doi.org/10.1109/ACCESS.2019.2918585.
[172] G. Li, M. Zeng, D. Mishra, L. Hao, Z. Ma, O.A. Dobre, Latency minimization for IRS-aided NOMA MEC systems with WPT-enabled IoT devices, IEEE Internet Things J. 10 (14) (2023) 12156–12168, https://doi.org/10.1109/JIOT.2023.3240395.
[173] D. Ye, X. Wang, J. Hou, Balanced multi-access edge computing offloading strategy in the Internet of things scenario, Comput. Commun. 194 (2022) 399–410, https://doi.org/10.1016/j.comcom.2022.07.048.
[174] O. Alamu, T.O. Olwal, K. Djouani, Cooperative NOMA networks with simultaneous wireless information and power transfer: an overview and outlook, Alex. Eng. J. 71 (2023) 413–438, https://doi.org/10.1016/j.aej.2023.03.057.
[175] K. Li, J. Zhao, J. Hu, Y. Chen, Dynamic energy efficient task offloading and resource allocation for NOMA-enabled IoT in smart buildings and environment, Build. Environ. 226 (2022) 109513, https://doi.org/10.1016/j.buildenv.2022.109513.
[176] K. Pahlavan, P. Krishnamurthy, Evolution and impact of Wi-fi technology and applications: a historical perspective, Int. J. Wireless Inf. Network 28 (1) (2021) 3–19, https://doi.org/10.1007/s10776-020-00501-8.
[177] A. Mahmood, A. Ahmed, M. Naeem, Y. Hong, Partial offloading in energy harvested mobile edge computing: a Direct search approach, IEEE Access 8 (2020) 36757–36763, https://doi.org/10.1109/ACCESS.2020.2974809.
[178] X. Yan, The application of power-domain non-orthogonal multiple access in satellite communication networks, IEEE Access 7 (2019) 63531–63539, https://doi.org/10.1109/ACCESS.2019.2917060.
[179] F. Chai, Q. Zhang, H. Yao, X. Xin, R. Gao, M. Guizani, Joint multi-task offloading and resource allocation for mobile edge computing systems in satellite IoT, IEEE Trans. Veh. Technol. 72 (6) (2023) 7783–7795, https://doi.org/10.1109/TVT.2023.3238771.
[180] S. Xia, Z. Yao, Y. Li, S. Mao, Online distributed offloading and computing resource management with energy harvesting for heterogeneous MEC-enabled IoT, IEEE Trans. Wireless Commun. 20 (10) (2021) 6743–6757.
[181] Y. Zhang, J.-H. Liu, C.-Y. Wang, H.-Y. Wei, Decomposable intelligence on cloud-edge IoT framework for live video analytics, IEEE Internet Things J. 7 (9) (2020) 8860–8873, https://doi.org/10.1109/JIOT.2020.2997091.
[182] R. Chai, M. Li, T. Yang, Q. Chen, Dynamic priority-based computation scheduling and offloading for interdependent tasks: leveraging parallel transmission and execution, IEEE Trans. Veh. Technol. 70 (10) (2021) 10970–10985, https://doi.org/10.1109/TVT.2021.3110401.
[183] C. Chen, Y. Zhang, Z. Wang, S. Wan, Q. Pei, Distributed computation offloading method based on deep reinforcement learning in ICV, Appl. Soft Comput. 103 (2021) 107108, https://doi.org/10.1016/j.asoc.2021.107108.
[184] L. Zhao, MESON: a mobility-aware dependent task offloading scheme for urban vehicular edge computing, IEEE Trans. Mobile Comput. (2023) 1–15, https://doi.org/10.1109/TMC.2023.3289611.
[185] S. Liu, Y. Yu, L. Guo, P.L. Yeoh, B. Vucetic, Y. Li, Adaptive delay-energy balanced partial offloading strategy in Mobile Edge Computing networks, Digit. Commun. Netw. 9 (6) (2023) 1310–1318, https://doi.org/10.1016/j.dcan.2022.05.029.
[186] B. Lu, S. Lin, J. Fang, X. Hong, J. Shi, Learning-assisted partial offloading for dynamic NOMA-MEC systems with imperfect SIC and reconfiguration energy cost, IEEE Internet Things J. 10 (22) (2023) 20134–20148, https://doi.org/10.1109/JIOT.2023.3283272.
[187] X. Li, R. Fan, H. Hu, N. Zhang, Joint task offloading and resource allocation for cooperative mobile-edge computing under sequential task dependency, IEEE Internet Things J. 9 (23) (2022) 24009–24029, https://doi.org/10.1109/JIOT.2022.3188933.
[188] H. Jiang, X. Dai, Z. Xiao, A. Iyengar, Joint task offloading and resource allocation for energy-constrained mobile edge computing, IEEE Trans. Mobile Comput. 22 (7) (2023) 4000–4015, https://doi.org/10.1109/TMC.2022.3150432.
[189] H. Hu, X. Zhou, Q. Wang, R.Q. Hu, Online computation offloading and trajectory scheduling for UAV-enabled wireless powered mobile edge computing, China Commun. 19 (4) (2022) 257–273, https://doi.org/10.23919/JCC.2022.04.019.
[190] F. Wang, S. Cai, V.K.N. Lau, Decentralized DNN task partitioning and offloading control in MEC systems with energy harvesting devices, IEEE J. Sel. Top. Signal Process. 17 (1) (2023) 173–188, https://doi.org/10.1109/JSTSP.2022.3221850.
[191] H. Zhou, M. Li, N. Wang, G. Min, J. Wu, Accelerating deep learning inference via model parallelism and partial computation offloading, IEEE Trans. Parallel Distr. Syst. 34 (2) (2023) 475–488, https://doi.org/10.1109/TPDS.2022.3222509.
[192] X. Jiao, Deep reinforcement learning for time-energy tradeoff online offloading in MEC-enabled industrial internet of things, IEEE Trans. Netw. Sci. Eng. (2023) 1–14, https://doi.org/10.1109/TNSE.2023.3263169.
[193] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, F. Bai, Hermes: latency optimal task assignment for resource-constrained mobile computing, IEEE Trans. Mobile Comput. 16 (11) (2017) 3056–3069, https://doi.org/10.1109/TMC.2017.2679712.
[194] L. Yang, B. Liu, J. Cao, Y. Sahni, Z. Wang, Joint computation partitioning and resource allocation for latency sensitive applications in mobile edge clouds, IEEE Trans. Serv. Comput. 14 (5) (2021) 1439–1452, https://doi.org/10.1109/TSC.2018.2890603.
[195] L. Yang, J. Cao, Z. Wang, W. Wu, Network aware mobile edge computation partitioning in multi-user environments, IEEE Trans. Serv. Comput. 14 (5) (2021) 1478–1491, https://doi.org/10.1109/TSC.2018.2876535.
[196] Y. Zhang, Resource scheduling and delay analysis for workflow in wireless small cloud, IEEE Trans. Mobile Comput. 17 (3) (2018) 675–687, https://doi.org/10.1109/TMC.2017.2734083.
[197] C.-S. Yang, R. Pedarsani, A.S. Avestimehr, Communication-aware scheduling of serial tasks for dispersed computing, IEEE/ACM Trans. Netw. 27 (4) (2019) 1330–1343, https://doi.org/10.1109/TNET.2019.2919553.

[198] "Joint Optimization of Offloading and Resource Allocation Scheme for Mobile Edge Computing | IEEE Conference Publication | IEEE Xplore." Accessed: January3, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8885537.
[199] S. Sundar, B. Liang, Offloading dependent tasks with communication delay and deadline constraint, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, 2018, pp. 37–45, https://doi.org/10.1109/INFOCOM.2018.8486305.
[200] L. Liu, H. Huang, H. Tan, W. Cao, P. Yang, X.-Y. Li, Online DAG scheduling with on-demand function configuration in edge computing, in: E.S. Biagioni, Y. Zheng, S. Cheng (Eds.), Wireless Algorithms, Systems, and Applications, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2019, pp. 213–224, https://doi.org/10.1007/978-3-030-23597-0_17.
[201] X. Qiu, L. Zhai, H. Wang, Time-minimized offloading for mobile edge computing systems, IEEE Access 7 (2019) 135439–135447, https://doi.org/10.1109/ACCESS.2019.2941825.
[202] R. Karthick, R. Ramkumar, M. Akram, M. Vinoth Kumar, Overcome the challenges in bio-medical instruments using IOT – a review, Mater. Today Proc. 45 (2021) 1614–1619, https://doi.org/10.1016/j.matpr.2020.08.420.
[203] L.P. Qian, B. Shi, Y. Wu, B. Sun, D.H.K. Tsang, NOMA-enabled mobile edge computing for internet of things via joint communication and computation resource allocations, IEEE Internet Things J. 7 (1) (2020) 718–733, https://doi.org/10.1109/JIOT.2019.2952647.
[204] B. Zhang, W. Ren, L. Zhao, X. Deng, Immune system multiobjective optimization algorithm for DTLZ problems, in: 2009 Fifth International Conference on Natural Computation, IEEE, 2009, pp. 603–607. https://ieeexplore.ieee.org/abstract/document/5365367/. (Accessed 24 October 2023).
[205] K. Zhang, S. Leng, Y. He, S. Maharjan, Y. Zhang, Mobile edge computing and networking for green and low-latency internet of things, IEEE Commun. Mag. 56 (5) (2018) 39–45, https://doi.org/10.1109/MCOM.2018.1700882.
[206] T.S. Lugovaya, "Biometric Human Identification Based on Electrocardiogram, Masters Thesis Fac. Comput. Technol. Inform. Electrotech. Univ. 'LETI' St.-Petersburg Russ. Fed., 2005.
[207] Y. Zuo, S. Jin, S. Zhang, Y. Zhang, Blockchain storage and computation offloading for cooperative mobile-edge computing, IEEE Internet Things J. 8 (11) (2021) 9084–9098, https://doi.org/10.1109/JIOT.2021.3056656.
[208] M. Andrew, Z. Denis, M. Crogan, Discriminators for use in flow-based classification, Queen Mary Westfield Coll. Dep. Comput. Sci. (2005).
[209] Z. Zhao, W. Zhou, D. Deng, J. Xia, L. Fan, Intelligent mobile edge computing with pricing in internet of things, IEEE Access 8 (2020) 37727–37735, https://doi.org/10.1109/ACCESS.2020.2974249.
[210] TPC-H, The TPC-H benchmarks [Online]. Available: www.tpc.org/tpch/, 2021.
[211] R. Yugha, S. Chithra, A survey on technologies and security protocols: reference for future generation IoT, J. Netw. Comput. Appl. 169 (2020) 102763, https://doi.org/10.1016/j.jnca.2020.102763.
[212] C. Sonmez, A. Ozgovde, C. Ersoy, EdgeCloudSim: an environment for performance evaluation of edge computing systems, Trans. Emerg. Telecommun. Technol. 29 (11) (2018) e3493, https://doi.org/10.1002/ett.3493.
[213] Z. Cai, Q. Li, X. Li, ElasticSim: a toolkit for simulating workflows with cloud resource runtime auto-scaling and stochastic task execution times, J. Grid Comput. 15 (2) (2017) 257–272, https://doi.org/10.1007/s10723-016-9390-y.
[214] H.H. Yang, Z. Liu, T.Q. Quek, H.V. Poor, Scheduling policies for federated learning in wireless networks, IEEE Trans. Commun. 68 (1) (2019) 317–333.
[215] J. Ren, Y. He, D. Wen, G. Yu, K. Huang, D. Guo, Scheduling for cellular federated edge learning with importance and channel awareness, IEEE Trans. Wireless Commun. 19 (11) (2020) 7690–7703.
[216] M. Zhang, et al., Communication-efficient federated edge learning via optimal probabilistic device scheduling, IEEE Trans. Wireless Commun. 21 (10) (2022) 8536–8551.
[217] D. Dholakiya, T. Kshirsagar, A. Nayak, Survey of Mininet challenges, opportunities, and application in software-defined network (SDN), in: T. Senjyu, P. N. Mahalle, T. Perumal, A. Joshi (Eds.), Information and Communication Technology for Intelligent Systems, Vol. 196, Smart Innovation, Systems and Technologies, vol. 196, Springer Singapore, Singapore, 2021, pp. 213–221, https://doi.org/10.1007/978-981-15-7062-9_21.
[218] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, A. Chaintreau, CRAWDAD Dataset Cambridge/haggle (v. 2006-09-15), CRAWDAD Wirel. Netw. Data Arch., 2006.
[219] L. Campanile, M. Gribaudo, M. Iacono, F. Marulli, M. Mastroianni, Computer network simulation with ns-3: a systematic literature review, Electronics 9 (2) (2020) 272.
[220] M. Chen, Y. Miao, I. Humar, Introduction to OPNET network simulation, in: M. Chen, Y. Miao, I. Humar (Eds.), OPNET IoT Simulation, Springer, Singapore, 2019, pp. 77–153, https://doi.org/10.1007/978-981-32-9170-6_2.
[221] L. P, et al., Eua dataset [Online]. Available: https://github.com/swinedge/eua-dataset, 2018.
[222] J. Grundy, Y. Yang, Optimal edge user allocation in edge computing with variable sized vector bin packing, Accessed: October24, 2023. [Online]. Available:, in: Service-Oriented Computing: 16th International Conference, ICSOC 2018, Hangzhou, China, November 12-15, 2018, Proceedings, Springer, 2018, p. 230 https://books.google.com.my/books?hl=en&lr=&id=sq52DwAAQBAJ&oi=fnd&pg=PA230&dq=Optimal+edge+user+allocation+in+edge+computing+with+variable+sized+vector+bin+packing&ots=7ENgTi4751&sig=N67KK_0ya6U90KjeDPm6ZLfm8Ao.
[223] A. Musa, I. Awan, Functional and performance analysis of discrete event network simulation tools, Simulat. Model. Pract. Theor. 116 (2022) 102470, https://doi.org/10.1016/j.simpat.2021.102470.
[224] J. Huang, M. Wang, Y. Wu, Y. Chen, X. Shen, Distributed offloading in overlapping areas of mobile-edge computing for internet of things, IEEE Internet Things J. 9 (15) (2022) 13837–13847, https://doi.org/10.1109/JIOT.2022.3143539.