A Systematic Review of Optimization Techniques for Computation Offloading in Mobile Edge Computing: Lyapunov, Convex, Heuristic, Game-Theoretic, and Machine Learning Approaches

Mohammad Ashique E Rasool[*1], Anoop Kumar [2], Asharul Islam [3]
[1, 2]Faculty of Mathematics and Computing, Banasthali Vidyapith, Rajasthan, Postal Code 304022, India
[3]College of Computer Science, King Khalid University, Abha, Postal Code 61421, Saudi Arabia
Email: ashique.rasool@gmail.com, anupbhola@banasthali.in, aaleslam@kku.edu.sa
[*]Corresponding author

Keywords: Computation offloading, edge computing, optimization methods, systematic review

Received: August 28, 2025

Mobile Edge Computing (MEC) has become a key paradigm for reducing latency, energy consumption, and system overhead in computation-intensive applications by enabling task offloading to edge servers. In this paper, we conduct a Systematic Literature Review (SLR) of 70 peer-reviewed studies, with 30 papers coded in detail across five main optimization approaches: Lyapunov, convex, heuristic, game-theoretic, and machine learning methods. The review systematically compares these paradigms in terms of optimization objectives, task assumptions, and evaluation setups. Our analysis reveals that while Lyapunov and game-theoretic approaches rely entirely on simulation (100%), heuristic studies exhibit stronger practical grounding, with 33% including real-world validation. Convex methods demonstrated 10–18% energy savings in heterogeneous MEC scenarios, whereas machine learning approaches showed adaptability under uncertain conditions but lacked real-world testing. Beyond numerical comparison, the review highlights the methodological evolution of MEC optimization, from formal mathematical programming to learning-driven and hybrid approaches. The coding of 30 representative papers provides a quantitative foundation that exposes common assumptions—such as independent task models and simplified simulators—that limit real-world applicability. At the same time, the synthesis identifies promising trends, including reinforcement learning for adaptive decision-making and hybrid frameworks that combine convex optimization with predictive models. These insights underscore the need for future MEC strategies that balance computational efficiency, adaptability, and scalability. By synthesizing results, quantifying trends, and framing open challenges, this SLR provides researchers and practitioners with a comprehensive understanding of the strengths, limitations, and opportunities in MEC computation offloading.

Povzetek: SLR o MEC odlaganju sintetizira 70 študij, primerja Lyapunovske, konveksne, hevristične in ML pristope, razkrije prevlado simulacij in poenostavitev ter izpostavi trend k RL ter hibridnim metodam za uravnoteženje učinkovitosti, prilagodljivosti in razširljivosti v praksi.

1 Introduction

The unprecedented growth in smartphone adoption, with over 5 billion users and 8.58 billion active mobile subscriptions worldwide, has driven a transformative shift in the digital ecosystem. This trend is amplified by the expected deployment of 55.7 billion IoT devices by 2025, which are anticipated to generate an overwhelming 80 zettabytes of data [1], [2]. Huawei projects an even more significant surge, estimating global data volumes will reach 181 zettabytes by 2025 [3]. This massive influx of data presents significant challenges for processing and utilization, necessitating innovative solutions to manage the demands of connected devices.

Cloud technology and data centers have become indispensable in handling this exponential data growth, with predictions indicating that 95% of new digital data will be processed on cloud infrastructures by 2025 [4]. Concurrently, mobile devices are increasingly hosting resource-intensive applications, including virtual and augmented reality, video streaming, image processing, and health monitoring, each with distinct and evolving computational requirements. Natural language processing tasks further add complexity, exhibiting sporadic and variable demands based on user interactions [5]. While quantum computing shows promise, its current capabilities are insufficient to manage the extensive data streams generated by mobile and IoT devices. Mobile Edge Computing (MEC) has emerged as a viable solution to address these challenges. By deploying resource-rich MEC servers closer to end devices, MEC improves application performance while reducing latency and energy consumption. This paradigm offers a practical approach to handling the computational demands of modern applications, creating a need for optimized offloading strategies to balance execution delay, energy

efficiency, and resource utilization. Figure 1 illustrates the MEC network architecture [6].

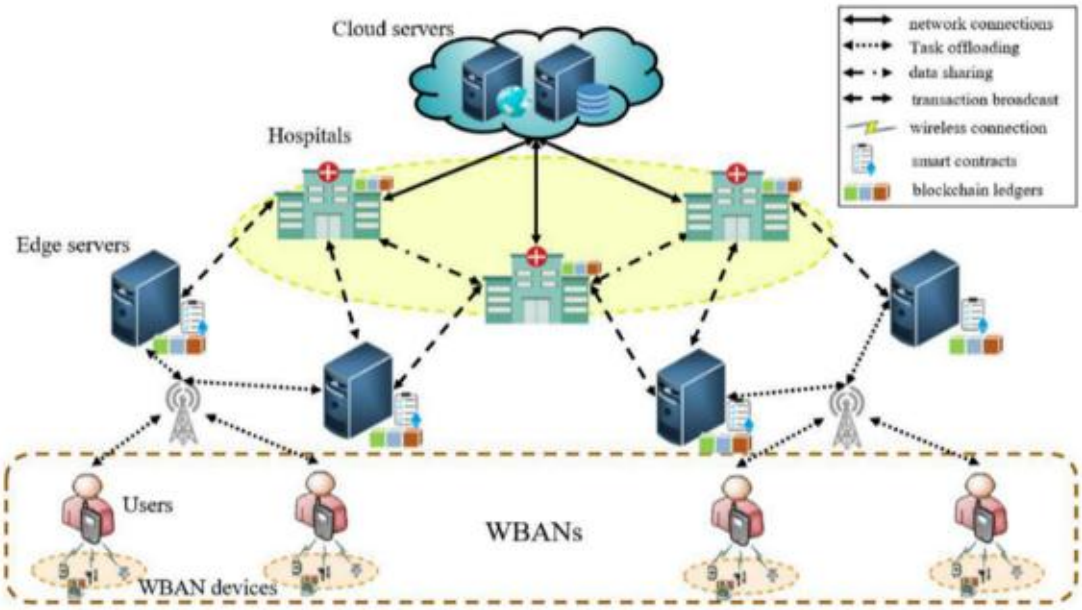

Figure 1: Depicts the network architecture of MEC environment [6]

This paper provides a detailed comparative analysis of computation offloading optimization techniques within MEC environments. It evaluates their strengths, limitations, and future potential, aiming to guide advancements in the field. The structure of the paper is as follows: Section 2 reviews related works to establish a foundation for the study. Section 3 introduces a taxonomy of offloading approaches, while Section 4 explores optimization techniques and algorithms. Section 5 presents a critical analysis of offloading optimization methods in Edge Computing, and Section 6 includes discussions, section 7 presents challenges and future research directions. Finally, Section 8 summarizes key findings and contributions.

2 Related work

This section reviews prior survey papers and related works in computation offloading and Mobile Edge Computing (MEC). The aim is to summarize existing surveys to highlight their scope, limitations, and the research gaps they leave unaddressed. By positioning our work against these earlier efforts, we clearly establish the novelty of our contribution, which is presented later in Section 5 as a comprehensive systematic analysis of the field. While previous studies have explored offloading decisions, resource allocation, and collaborative optimization, a comprehensive synthesis is still lacking. This review addresses the gap by systematically examining optimization techniques within EC frameworks, serving as a foundational reference for researchers and practitioners.

Zhang and Shi [7] explored allocation and management schemes in mobile edge networks and demonstrated improvements in throughput and system utility under varying user densities. More recently, Zhang [8] proposed a multi-objective genetic algorithm with knowledge-based crossover and segmentation mutation, achieving significant gains in energy consumption, system utilization, and task completion rates.

Mach and Becvar [9] investigated Multiaccess Edge Computing (MEC) integration, focusing on computational task offloading and resource allocation for efficient task distribution. Similarly, Lin et al. [10] explored task partitioning and execution challenges in EC, emphasizing the importance of optimizing task distribution to enhance performance. Zheng et al. [11] provided a concise survey of common scenarios and strategies for offloading in edge computing.

Shakarami et al. contributed significantly through two reviews: one addressing game-theoretic approaches to collaborative optimization [12], and another focusing on machine learning (ML) methods to enhance system performance [13]. Their work highlights the potential of ML in refining offloading processes. Nguyen and Dressler [14], examined cooperative communication systems, comparing algorithms but without delving into their computational complexities.

The growing focus on EC and offloading optimization has resulted in a wealth of innovative solutions, yet gaps remain. Existing reviews often target specific methodologies, leaving a need for a holistic analysis of optimization methods tailored to computation offloading.

To clarify the novelty of our work compared to prior reviews, we summarize the scope, strengths, and limitations of previous surveys in Table 1. Unlike these earlier efforts, which typically focused on a single optimization paradigm or limited domain, our work

provides a holistic, PRISMA-based systematic review of 71 studies covering Lyapunov, convex, heuristic, game-theoretic, and machine learning approaches. This broader scope and methodological rigor clearly distinguish our contribution.

Table 1: Compression of previous work

Study	Scope / Focus	Strengths	Limitations	Contribution of This Work
[15]	Focus on game-theoretic models for MEC offloading	Clear categorization of equilibrium-based methods	Narrow focus on game theory; ignores heuristics and ML-based methods	Our work covers multiple paradigms (Lyapunov, convex, heuristics, game theory, ML)
[16]	Energy-efficient MEC optimization using heuristics	Strong treatment of heuristic techniques	Does not address formal optimization or machine learning; limited dataset	We provide a systematic comparison across 71 papers, including formal and learning-based approaches
[17]	Survey on convex and Lyapunov optimization in MEC	Rigorous mathematical insights	Overlooks heuristic and data-driven methods; no empirical coding of papers	Our work integrates formal optimization with heuristic and ML trends, bridging the gap
[18]	MEC offloading in IoT contexts	Good discussion of IoT-specific architectures	Lacks generalizability; not comprehensive beyond IoT	Our work provides a holistic view of MEC offloading across domains
This Work	Systematic Literature Review of 74 studies, analyzing multiple optimization paradigms	Comprehensive scope; PRISMA-based methodology; quantitative coding of 30 papers	-	Holistic, systematic SLR covering multiple optimization paradigms, with explicit identification of trends, challenges, and gaps

This review aims to bridge this gap, offering a consolidated resource for understanding the current state of research, identifying gaps, and guiding future investigations.

Key contributions of this review include:

- *Exploration of Communication and Computation Models:* Providing foundational insights into models relevant to computation offloading.
- *Examination of Optimization Methods:* Analyzing strengths, limitations, and applicability of optimization techniques across various domains.
- *Identification of Research Issues:* Highlighting unexplored areas to inspire further research into emerging challenges and opportunities.

This review scrutinizes the optimization challenges in EC, categorizing methodologies into widely applied techniques like Lyapunov optimization [19], [20], convex optimization [21], [22], heuristic optimization [23], game theory [24], and machine learning based optimization techniques [25], [26].

By presenting a consolidated analysis, this review provides a pivotal resource for researchers and practitioners to navigate the intricate domain of computation offloading in EC environments efficiently.

To complement this comparison with prior surveys, the detailed attributes of the papers reviewed in this SLR are presented in Tables 3–7. These tables summarize optimization methods, evaluation criteria, task assumptions, and experimental environments (simulation or real) for the 30 coded studies. Representative performance outcomes (e.g., latency reduction, energy savings, and adaptability) are further synthesized in Section 5 to highlight the distinct contributions and gaps in the current state-of-the-art.

3 Taxonomy

3.1 Computation offloading

Computation offloading is a crucial strategy for enhancing the performance and user experience of resource-constrained mobile devices by transferring tasks to more capable computing resources. These resources may include distant cloud servers, edge computing servers within wireless networks, or even nearby mobile devices. Offloading is categorized into two types: binary offloading, where the entire task is offloaded, and partial offloading, where only portions of the task are transferred. The choice depends on the specific requirements of the application.

Mobile applications typically generate two types of tasks: atomic tasks and compound tasks. Atomic tasks, being indivisible, are either executed entirely on the source device or fully offloaded to a resource-rich device. In contrast, compound tasks are divisible into subtasks, enabling partial offloading [27] This allows certain

subtasks to be executed concurrently across multiple resources, offering flexibility and efficiency.

Optimizing offloading decisions involves striking a balance to minimize delay, power consumption, and cost. Simply deploying high-performance edge servers at telecom access points does not guarantee optimal outcomes. While edge servers excel at handling computation-intensive tasks, large data sizes can lead to increased transmission times. Moreover, the dynamic nature of wireless channels and the variability in computation resource availability further complicate the decision-making process.

To address these challenges, researchers have developed computation models and optimization techniques aimed at identifying optimal offloading strategies. These approaches focus on leveraging the advantages of edge computing to improve task execution efficiency while overcoming limitations posed by mobile devices.

The following sections of this study comprehensively review these advancements, highlighting state-of-the-art methods and their impact on addressing offloading challenges in edge computing environments.

3.2 Computation models

Computation models estimate execution delay based on task workload and CPU frequency, transmission delay on data size and channel rate, and energy consumption on CPU cycles and transmission power.

In practice, these models are applied differently across optimization methods: Lyapunov-based works [46-51] use them to maintain queue stability under dynamic workloads, convex optimization studies [52-55] apply them for minimizing total energy consumption, heuristic approaches [56-61] extend them to workflows with dependent tasks, game-theoretic methods [32], [36], [62-65] use them to balance resource allocation among competing users, and ML-based techniques [66-71] employ them to capture environment uncertainty during adaptive decision-making.

Considering a set of M mobile devices represented as: $U=\{D_1, D_2, D_3, \ldots\ldots, D_M\}$, connected to remote resource wirelessly. Each device having N tasks represented as $T=\{T_{m,1}, T_{m,1}, T_{m,1}, \ldots., T_{m,N},\}$. Each mobile device Dm is characterized by two distinct parameters: processing power (p_m) and transmission power (t_m). The processing power (p_m) signifies the computational capacity of the device, determining its ability to execute tasks and handle computational workloads efficiently. On the other hand, the transmission power (t_m) represents the energy consumed by the device during data transmission, reflecting its communication capabilities and energy usage for transmitting data over wireless networks. Every task Tm,n is characterized by two essential parameters: Dm,n and $C_{m,n}$. The former denotes the input data size required for the task $T_{m,n}$, while the latter signifies the number of CPU cycles necessary to execute the task $T_{m,n}$.

A mobile device executes tasks locally if resources are sufficient, otherwise offloads them to a MEC server, with total task time comprising local execution, offloading, and MEC processing phases, while energy costs arise from either local execution or transmission [28].

3.2.1 Local computing

In local computing the time needed to complete the task $T_{m,n(local)}$ defined in Eq(1) depends upon both the needed CPU cycles, $C_{m,n}$ and frequency Fm of the mobile device [29-34].

$$T_{m,n(local)} = \frac{C_{m,n}}{F_m} \tag{1}$$

The energy required by the mobile device to execute task $T_{m,n(local)}$ defined in Eq(2) is represented as follows [27], [31-33]:

$$E_{m,n(local)} = \kappa C_{m,n} F_m^2 \tag{2}$$

The switched capacitance coefficient (κ) plays a crucial role in chip architecture, influencing various aspects of its performance. Authors have proposed different values for κ, as documented in table 2. For a deeper understanding of the effective switched capacitance, readers can refer to Ref. [38].

Table 2: Values of κ

Reference Work	Values of κ
[24], [31]	10^{-11}
[26], [35]	10^{-26}
[33], [36]	10^{-27}
[25], [37]	10^{-28}

3.2.2 Task offloading

The wireless offloading rate R_m represented in Eq(3) via an Access Point depends on bandwidth, device transmission power, distance, and interference, and is formally defined using the Shannon–Hartley theorem [28], [42].

$$R_m = \mathrm{B}\log_2\left(1 + \frac{E_m G_m}{\sigma + \sum_{i \neq \mathrm{m}, j \neq \mathrm{m}} P_{i,j} G_{i,j}}\right) \tag{3}$$

Where B is transmission channel bandwidth, Em denotes transmission energy of D_m required for uploading a task to the edge server via access point, G_m denotes channel gain in the U_m and the respective access point during task transmission. $G_m = \left(dis_{m,a}^{-\eta}\right)|h_{m,a}|^2$, where $dis_{m,a}^{-\eta}$ denotes the loss effect of the communication channel, $dis_{m,a}$ represents Euclidean distance of D_m and the access point, η is the exponent of communication channel loss[41], $h_{m,a}$ is the refers to the Raileigh fading channel coefficient, which follows a normal distribution with a mean of 0 and a variance of 1 and σ represents the channel noise. In most cases, when a task is sent from an SMD to an MEC server, the time it takes depends mainly

on wireless communication. This is because wired channels have very high bandwidth, making their transmission time almost insignificant. Therefore, the transmission time for $T_{m,n}$ is calculated as follows [31], [33], [34], [44].

$$t_{m,n}^{tr} = \frac{d_{m,n}}{R_m} \quad (4)$$

Where $t_{m,n}^{tr}$ Eq(4) represents the time elapsed in receiving the input data $T_{m,n}$ at MEC server from the device D_m.

The energy used $E_{m,n}^{tr}$ in Eq(5) by device D_m to offload task $T_{m,n}$ to the edge server is calculated in [30], [31], [37] as follows:

$$E_{m,n}^{tr} = p_m t_{m,n}^{r} \quad (5)$$

Where p_m represents transmission power of the mobile device D_m. When a task comes in, the MEC server provides computing power to the connected device to help it finish the task by assigning a computing clock frequency $f_{m,n}^{e}$. In this way, the task is completed at the MEC serve according to the following model [30], [34], [44]:

$$t_{m,n}^{MEC} = \frac{c_{m,n}}{f_{m,n}^{e}} \quad (6)$$

Where $\sum_{m=1}^{M}\sum_{n=1}^{N} O_{m,n} f_{m,n}^{e} \leq F$, this means the total computing power allocated cannot be more than the MEC server's available computing capacity. The mean execution time represented as T in Eq(7) and power consumption to execute the task represented as E in Eq(8) are identified in [31].

$$T = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left(1 - O_{m,n}\right)t_{m,n}^{loc} + O_{m,n}\left(t_{m,n}^{tr} + t_{m,n}^{MEC}\right) \quad (7)$$

$$E = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left(1 - O_{m,n}\right)E_{m,n}^{loc} + O_{m,n}\, E_{m,n}^{tr} \quad (8)$$

Here, $O_{m,n} \in \{0,1\}$ is a binary variable. It is set to 0 when the task is done locally and 1 when the task is executed elsewhere. Therefore, we face a multi-objective optimization problem to minimize, as explained in [31].

$$P1: \min_{O_{m,n}} \{E, T\}$$

s.t. (9)

$$C1: O_{m,n} \in \{0,1\}$$
$$C2: \sum_{m=1}^{M}\sum_{n=1}^{N} O_{m,n} f_{m,n}^{e} \leq F$$

One can add extra conditions in the above formulation, like ensuring that the delay is less than each task's deadline. To simplify this multi-objective optimization problem, one can combine all the objectives into one objective by assigning a weight to all objectives under consideration [28), 31), 32), 36), 45)].

$$Z = \gamma T + (1 - \gamma)E \quad (10)$$

$$P1: \min_{O_{m,n}} \{Z\}$$

s.t. (11)

$$C2: \gamma \in [0,1]$$
$$C3: \sum_{m=1}^{M}\sum_{n=1}^{N} O_{m,n} f_{m,n}^{e} \leq F$$

Where, value of γ in which ranges between 0 and 1, depends on the specific task. For instance, γ is set closer to 1 for tasks that prioritize speed and closer to 0 for applications that focus on saving energy.

4 Research methodology

This study employs a Systematic Literature Review (SLR) methodology to comprehensively analyze and evaluate optimization methods in Mobile Edge Computing (MEC) environments. The SLR approach ensures a structured and rigorous synthesis of existing research, providing a foundation for identifying strengths, limitations, and future research opportunities. The following steps outline the methodology:

4.1 Research objectives and questions

The primary objectives of this study are:

- To evaluate the effectiveness of various optimization methods in MEC offloading, including Lyapunov optimization, convex optimization, heuristic techniques, game theory, and machine learning.
- To analyze the computational complexities, applicability, and outcomes of these methods.
- To identify research gaps and propose directions for future investigation.

Key research questions include:

1. What are the strengths and weaknesses of existing optimization methods in MEC offloading?
2. How do these methods address challenges such as energy efficiency, latency, and task allocation?
3. What are the current gaps and limitations in MEC optimization research?

4.2 Inclusion and exclusion criteria

Inclusion criteria:

- Peer-reviewed articles, conference proceedings, and technical reports focusing on MEC offloading.
- Studies analyzing Lyapunov, convex, heuristic, game-theoretic, and machine learning optimization techniques.
- Papers providing detailed discussions on computational complexities, experimental setups, and outcomes.

Exclusion criteria:

- Articles not directly related to MEC offloading.
- Studies lacking empirical data or focused solely on theoretical frameworks without implementation insights.

4.3 Search strategy

A comprehensive search was conducted across leading scientific databases, including IEEE Xplore, ACM Digital Library, SpringerLink, and ScienceDirect. Keywords and Boolean operators were used to ensure exhaustive coverage, such as:

- "Mobile Edge Computing" AND "computation offloading"
- "optimization methods" AND "Lyapunov" OR "convex" OR "heuristic"

• “machine learning” AND “offloading optimization”

Reference lists of included articles were further scanned for relevant studies to ensure no critical research was overlooked.

The study selection process, including identification, screening, eligibility, and inclusion of studies, was conducted in accordance with PRISMA 2020 guidelines and is summarized in Figure 2.

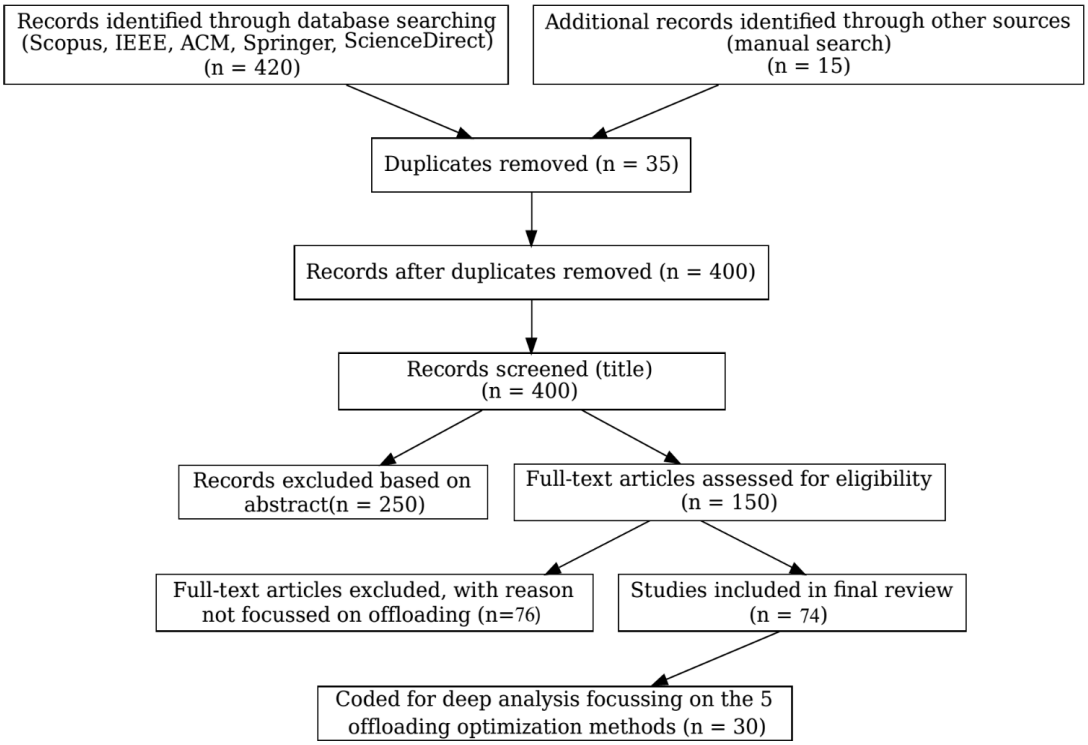

Figure 2: PRISMA flow diagram illustrating the identification, screening, eligibility, and inclusion for this SLR

4.4 Data extraction and synthesis

From each selected study, the following data were extracted:

- **Optimization criteria:** Objectives such as latency reduction, energy efficiency, or cost minimization.
- **Offloading methods:** Binary or partial offloading strategies.
- **Algorithm details:** Computational complexity, experimental setups, and key results.
- **Task dependency:** Whether the tasks were independent or interdependent.

The extracted data were systematically synthesized and categorized to facilitate a detailed comparative analysis.

4.5 Analysis and categorization

The selected studies were categorized based on the optimization method employed:

1. Lyapunov Optimization
2. Convex Optimization
3. Heuristic Techniques
4. Game Theory
5. Machine Learning

Each category was critically analyzed to identify patterns, challenges, and opportunities. Comparative metrics such as computational complexity, scalability, and practical applicability were used to evaluate these methods.

5 Analysis of optimization methods

In this section, we delve deeper into some of recent solutions for optimizing cost optimization problems. The solutions are grouped based on the techniques used to solve them and finally we discuss the strengths and weaknesses of these MEC optimization methods based on our analysis.

To structure the comparative analysis, the five major optimization paradigms applied in MEC offloading—Lyapunov, convex, heuristic, game-theoretic, and machine learning—are organized into a taxonomy, as illustrated in Figure 3. This taxonomy highlights the primary focus of each method and provides a visual roadmap for the detailed discussion that follows.

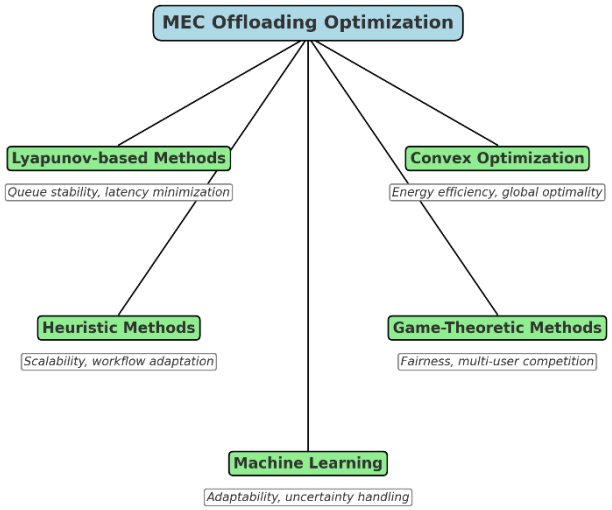

Figure 3: Taxonomy of optimization approaches in MEC offloading

5.1 Applying Lyapunov optimization

This, one of the well-known optimization methods used in [46] in the perspective of service providers. The challenge is deciding where and when to migrate services to reduce operational costs. Instead of relying on traditional programming methods that require statistical expertise, the authors suggested a different approach. They framed an objective problem as a successive decision-making task using Markov decision processes and developed a dynamic algorithm with Lyapunov optimization method. This algorithm doesn't need statistical information and has a complexity order of O(|H|²Tf), where H represents all possible task configurations, and Tf is the frame size.

Guo et al. [47] proposed the LOMUCO algorithm for Mobile Edge Computing (MEC) systems, designed to optimize partial computation offloading for multiple mobile devices (MDs) equipped with energy harvesting modules. Their goal was to minimize the long-term energy consumption of MDs while satisfying task delay constraints. The algorithm operates over discrete time slots and has a computational complexity of $O(T \cdot N^3)$ where T represents the number of time slots and N denotes the number of mobile devices.

Katsalis et al. [48] presented an approach to improve scheduling in MEC-IaaS environments by employing the Lyapunov optimization method. This method aims to enhance the revenue of MEC-IaaS providers while reducing violations of service level agreements (SLAs) for deployed services. The algorithm's complexity is expressed as $(O(|V||S||M||J|)^{3.5}L)$, in which V denotes virtual machines, S represents services, M refers to edge-cloud physical machines, J indicates mobile operators, and L corresponds to the input bits size.

Lyu et al. [49] developed an algorithm for Mobile Edge Computing (MEC) systems with thousands of IoT devices. Their goal was to increase the system's data processing efficiency while ensuring fairness among smart mobile devices (SMDs). The algorithm they proposed had a time complexity of O(MlogM), where M represents the number of IoT devices.

Xia et al. in [50] introduced a dynamic and distributed cost optimization algorithm designed for an environment with varied number of servers and different types of IoT applications. The order of their algorithm was calculated as $O(\Sigma_s \Gamma_{is})$, in which Γ_{is} represents the interactions between a mobile device m and a server s.

In [51] Lie and co-authors used Lyapunov optimization method for boosting processing speed in wirelessly powered mobile edge environment. To keep the process fast, they designed an algorithm with an execution order of O(M), in which M represents the number of devices. Table 3 shows comparison of offloading optimization leveraging Lyapunov optimization method in MEC Environment.

Table 3: Comparison of offloading optimization based on Lyapunov optimization method

Ref	Optimization Criteria	User Setup	Offloading Method	Task Dependency	Experimental Setup	Algorithm Complexity
[46]	To minimize task failure and latency	Multiuser	Binary offloading	Independent	Simulator and real dataset both	$O(\|H\|^2T_f)$
[47]	To minimize power consumption and latency	Multiuser	Partial offloading	Independent	Simulator	$O(T \cdot N^3)$
[48]	To minimize Overall cost	Multiuser	Binary offloading	Independent	Simulator	$(O(\|V\|\|S\|\|M\|\|J\|)^{3.5}L)$
[49]	To maximize profit	Multiuser	Partial offloading	Independent	Matlab Simulator	O(MlogM)
[50]	To maximize edge server utility	Multiuser	Partial offloading	Independent	Simulator	$O(\sum_s \Gamma_{is})$
[51]	To maximize computation rate	Multiuser	Partial offloading	Independent	Simulator	O(M)

Overall, Lyapunov-based approaches dominate latency-sensitive studies by providing queue stability and bounded delay guarantees, but they are rarely validated in real-world environments. Their reliance on cubic or higher-order complexity (e.g., O(|H|²Tf)) limits scalability to larger multi-user networks, making them less suitable for

practical deployment despite strong theoretical foundations.

5.2 Applying convex optimization

Due to the nonconvex nature of the constraints, most computation offloading problems are inherently nonconvex in nature. The nonconvex problems are classified as NP-hard, making them hard to solve. For achieving a polynomial time complexity solution, various techniques are employed to convert the nonconvex problems into convex one. These techniques include the reformulation linearization method, the Sequential Convex Programming (SCP) approach, and the Variable Substitution method. Once transformed, these problems are addressed by Newton's barrier method, and Lagrange techniques.

Mao et al. [52] employed convex-optimization along with flow-shop scheduling and techniques to minimize delay and power utilization in a single-user Mobile Edge Computing environment. Their objective function incorporated both integral variable values and continuous variables, representing resource allocation and transmission power allocation decisions, respectively. As a result, the problem was classified as mixed-integer nonlinear problem. Due to computational difficulty in exhaustive search methods, researchers introduced an alternative trivial algorithm with reduced complexity of order O(NlogN), in which N represents number of independent tasks.

Convex optimization approach is being extensively applied to address energy consumption minimization challenges in many research works. You and co-authors in [53] explored resource scheduling problem in a MEC environment with multiple users. Their study optimized the offloading strategy on the basis of channel gains and power consumption at the local device while adhering to time constraints. The proposed algorithm exhibited a complexity of O(Mlog(1/ϵ)), in which M represents number of mobile devices, with ϵ denoting accuracy of the solution. Although the study provided detailed modeling analyses focused on user offloading proportions, it did not consider the combined optimization of channel resources and compute resources.

Table 4: Comparison of offloading methods in MEC environment using convex optimization method

Ref	Optimization Criteria	User Setup	Offloading Method	Task Dependency	Experimental Setup	Algorithm Complexity
[52]	To reduce power consumption and latency	Singleuser	Partial Offloading	Independent	Simulator	O(NlogN)
[53]	To reduce power consumption	Multiuser	Binary Offloading	Independent	Simulator	O(Mlog(1/ϵ))
[35]	To reduce power consumption and latency	Multiuser	Partial Offloading	Independent	Experiment in real scenario	$O(N \cdot k_{max} \cdot k_{power})$
[34]	To minimize latency	Multiuser	Partial Offloading	Independent	Simulator	$O(M(K+2)+1)^x k$
[54]	To minimize system usage and cost	Multiuser	Binary Offloading	Independent	Simulator	$O((2M1+M2)^x k)$
[55]	To reduce power consumption	Multiuser	Partial Offloading	Independent	Simulator	O(MlogM)

In study [35], the researchers applied convex optimization techniques for addressing the multi-objective optimization problem. Order of algorithms in [29] was $O(N \cdot k_{max} \cdot k_{power})$, in which N represents the task count, k_{max}, is the iterations, and k_{power} refers to transmission power adjusting iterations using the Newton iteration method. They introduced an online offloading and resource allocation strategy designed to minimize power consumption while meeting deadline constraints and the requirements related task dependency. They proposed a distributed eDors algorithm to solve this problem.

The Alternating Direction Method of Multipliers (ADMM) has emerged as a central technique for solving optimization problems [34] and [54], both studies examined a resource scheduling problem aimed at minimizing latency and maximizing system utilization in Mobile Edge Computing environment. Their algorithms complexity was comparable. Specifically, the order of the algorithm proposed in [34] is $O(M(K+2)+1)^x k$, in which M depicts edge nodes count available in the next level bigger base station, whereas K is referring device to device links, x>1 indicates a polynomial-time algorithm, whereas k represents iterations. Tan and co-authors in [54] addressed a resource scheduling issue in full duplex cell networks that incorporate Mobile Edge Computing and caching. To tackle this issue, the researchers employed ADDM theory. Their algorithm's complexity is $O((2M1+M2)^x k)$, where M1 represents the number of client devices requiring service I, whereas M2 denotes the number of clients requiring service II.

In [55] the researchers focused on resource allocation for uplink communication in both full and partial offloading modes. They developed a customized greedy search algorithm to handle indivisible tasks and a low-complexity algorithm for divisible tasks to efficiently find solutions. To overcome the limitations of traditional

schemes like TDMA and OFDMA, the authors proposed a multiple access strategy. The order of their algorithm is O(MlogM), in which M represents the number of users. Table 4. shows Comparison of offloading optimization based on convex optimization method in MEC Environment.

Convex optimization methods are among the most widely applied, reflecting their dominance in energy-efficiency studies. They consistently achieve measurable savings (10–18% in our corpus) but are constrained by assumptions of complete system knowledge. Their polynomial-time complexity (e.g., O(MlogM)) makes them moderately scalable, though solver overhead remains a challenge in dynamic MEC environments.

5.3 Applying heuristics optimization

In situations where offloading decisions must be made quickly, heuristic techniques often serve as practical and effective solutions, even though they may not always produce optimal results. A notable advantage of heuristic methods is their ability to function efficiently when combined with other approaches. This subsection highlights key implementation of heuristic methods in addressing computation offloading problems. A comparison of various heuristic-based techniques is summarized in table 5.

In Ref. [56], a method based on cache-aware task scheduling was introduced for edge computing environment. The researchers formulated a resource scheduling problem as a assignment problem graph, where the weights of the selected parameters were determined primarily by the location of required data, like cache, disk, edge or cloud storage. The proposed solution involved finding the maximum weighted match among the executing tasks and computational resources. The computational order of their algorithm is $O([(N/Nc)].N^3)$, in which N represents the task count and Nc denotes the containers count.

Li et al. [57] presented an optimal resource scheduling strategy aimed at minimizing the financial expenses associated with rented edge nodes. The algorithm's performance was assessed using a real-world dataset, demonstrating a notable reduction in financial costs compared to existing algorithms. The computational order of the proposed solution was determined to be O(Imjk), in which I, m, j, and k represent edge nodes count, types of nodes, cloud nodes, and iterations, respectively.

In [58], the researchers explored task scheduling optimization for cyber-physical applications within a fog environment, emphasizing the reduction of total execution time. Their approach utilized a scheduler derived from moth-flame optimization method.

Similarly, Yang and co-authors in [59] introduced a hybrid solution combining the existing ASO and Pro-ITGO offloading algorithms, for minimizing power consumption over strict deadline constraints. By leveraging the strengths of edge computing and cloud computing paradigms, they suggested an integrated architecture. The order of their algorithm is calculated as O(N*I), in which N represents the task count, and I denotes the edge servers count.

In [60], the researchers address two sub-problems within a fixed offloading decision framework: fog computing resource allocation and user transmission power allocation. These issues are tackled using convex optimization techniques and Karush- Kuhn-Tucker (KKT) conditions. To enhance the simulated annealing (SA) algorithm, a memory function is integrated into its inner loop, resulting in the novel N-SA algorithm. Simulation results show that N-SA efficiently achieves optimal solutions and reduces system costs by 17% compared to the greedy offloading and joint resource allocation (GO-JRA) algorithm. In [61], The study introduces two LSTM-based prediction methods for decision-making in task offloading. In scenarios with limited resources, these methods enhance resource utilization by involving edge devices in offloading tasks, ensuring timely completion of latency-sensitive tasks and enabling predictive decision-making.

Table 5: Comparison of offloading methods in MEC environment using heuristic optimization method

Ref	Optimization Criteria	User Setup	Offloading Method	Task Dependency	Experimental Setup	Algorithm Complexity
[56]	To minimize power consumption and latency	Singleuser	Binary Offloading	Dependent	Hadoop, Java based experiment with real data	$O([(N/Nc)].N^3)$
[57]	To minimize cost	Singleuser	Binary Offloading	Independent	Hadoop, Java based experiment with real data	O(nmjk)
[58]	To minimize latency	Multiuser	Partial Offloading	Independent	iFogSim Simulator	N/A
[59]	To minimize power consumption	Singleuser	Partial Offloading	Independent	Workflowsim Simulator	O(N*I)
[60]	To reduce system cost	Multiuser	Partial Offloading	Independent	Matalb based silulator with actual data	O(T·O(f)·S)
[61]	To minimize latency	Multiuser	Partial Offloading	Independent	Simulator	O(T·N·D)

For resource-abundant situations, a polynomial-time optimal mechanism is proposed for pre-emptive offloading decisions. By addressing the 0–1 knapsack problem, the approach effectively meets the requirements of low-latency tasks. Numerical results validate the method's effectiveness.

Heuristic methods are the most adaptable to diverse contexts, including dependent workflows, and show the strongest real-world validation among the categories (33% of coded studies). Their reported complexities are generally polynomial, sometimes cubic (e.g., $O(N^3)$, $O(T \cdot N \cdot D)$), which means they are less scalable than linear or logarithmic-time methods, but still more practical than the very heavy formulations sometimes associated with Lyapunov-based optimization. While they provide responsiveness and flexibility, heuristics generally trade off global efficiency for computational feasibility in moderately sized MEC systems.

5.4 Game theory based optimization

Game theory, often referred to as a fundamental theory that examines the strategies employed by more than one decision parameter. It is widely applied to address problems in almost all scientific and social sciences. A comparison of various Game Theory based optimization techniques is summarized in table 6.

Zhang et al. [32] applied game theory to solve the joint optimization problem of computation offloading and resource allocation in heterogeneous networks. Order of their algorithm is $O(M_c \cdot k \cdot M_l)$, where M_c represents the set of offloading mobile terminals, M_l refers to the set of client devices considered for local computation, whereas k denotes iteration count.

The authors in Ref. [36] noted that most existing research has concentrated on single-carrier NOMA. To address this, they suggested a solution for combinatorial optimization problem for multicarrier NOMA-based mobile edge computing systems, derived from coalition game theory. Order of their suggested algorithm is O(k), in which k represents iterations count.

Currently, various application types, like virtual reality, augmented reality and video streaming, image processing, health applications, etc. are widely used, each having distinct resource requirements. To enhance Quality of Service (QoS), it is important to consider the specific needs of these heterogeneous tasks. In [62], the researchers proposed an algorithm based on cooperative game theory for improving the efficiency of heterogeneous tasks in Multi-access Edge Computing (MEC), focusing on latency. The execution order of their suggested algorithm is $O((I+1)^{|M|})$, in which I represent edge servers count, and M represents client devices count.

In [63], they proposed a coarse-grained offloading mechanism and a smart resource pairing policy within the framework of Cloud-Edge collaboration to effectively utilize cloud and edge server resources. By accounting for the heterogeneity of mobile devices and inter-channel interference, they formulated the task offloading decisions of multiple end-users as a game-theory-based task migration model aimed at maximizing system utility. Furthermore, they introduced an enhanced game-theory-based particle swarm optimization algorithm to determine the task offloading strategies. Experimental results demonstrate that their proposed scheme outperforms alternative approaches in terms of latency and energy consumption and exhibits good scalability as the number of mobile devices increases. Given these factors, the time complexity of the improved PSO algorithm can be expressed as: $O(P \cdot I \cdot O(f))$ where P is the number of particles in the swarm, I is the number of iterations, and O(f) is the complexity of evaluating the objective function at each iteration.

To develop an effective task offloading strategy, the combination of Multi-Access Edge Computing (MEC) with artificial intelligence has emerged as a promising solution. In this regard, [64] proposed a task offloading decision mechanism (TODM) that leverages cooperative game theory and deep reinforcement learning (DRL). A joint optimization problem is formulated with the goal of minimizing both the overall task processing delay (OTPD) and overall task energy consumption (OTEC).

Table 6: Comparison of offloading methods in MEC environment on game-theory based optimization method

Ref	Optimization Criteria	User Setup	Offloading Method	Task Dependency	Experimental Setup	Algorithm Complexity
[32]	To minimize latency and power consumption	Multiuser	Binary offloading	Dependent	Simulator	$O(M_c \cdot k \cdot M_l)$
[36]	To minimize latency and power consumption	Multiuser	Binary offloading	Dependent	Simulator	O(k)
[62]	To minimize latency	Multiuser	Partial offloading	Independent	Simulator	$O((I+1)^{\|M\|})$
[63]	To reduce delay and power consumption	Multiuser	Partial offloading	Dependent	Simulator	
[65]	To reduce latency and task failure	Multiuser	Partial offloading	Dependent	Simulator	$O(T \cdot N \cdot M \cdot f_{game})$
[64]	To reduce delay and power consumption	Multiuser	Partial offloading	Dependent	Simulator	$O(N \cdot f_{game} + E \cdot S \cdot A)$

The approach treats task vehicles (TaVs) and service vehicles (SeVs) as participants in a cooperative game, collaboratively designing offloading strategies to optimize resource utilization. The algorithm complexity is $O(N \cdot f_{game} + E \cdot S \cdot A)$, where N is the number of participants, f_{game} represents the complexity of solving the cooperative game, and E, S, and A are the number of episodes, states, and actions.

[65] proposes a multi-tier Vehicle-to-Everything (V2X) network framework that integrates vehicle nodes (VNs), mobile edge computing (MEC) servers, and cloud servers for computation offloading. The problem is formulated as a game, and a joint optimization algorithm, JOCORA, is introduced to achieve Nash equilibrium, improving offloading success and reducing latency. Simulations show JOCORA outperforms other methods. The algorithm complexity is projected to be $O(T \cdot N \cdot M \cdot f_{game})$ where, T represents simulation iterations count or episodes, N denotes vehicles count, M represents servers count (VNs, MEC, cloud), f_{game} is the complexity of calculating the Nash equilibrium or solving the game.

Game-theoretic approaches address fairness and multi-user resource competition effectively, but their applicability is limited by reliance on idealized assumptions and complete rationality of participants. Their complexities, often linear or polynomial (e.g., $O(M^c \cdot k \cdot M^l)$), are scalable, yet the lack of real-world validation reduces their immediate relevance.

5.5 Machine learning based optimization

Machine Learning (ML) is anticipated to replace conventional optimization techniques [59].in particular, deep learning (DL), a subset of ML, has recently been explored as a promising approach for optimizing computation offloading.

In Ref. [66], the authors suggested a deep Q-Learning model to achieve optimal scheduling in time-critical systems in terms of energy efficiency. The scheme is capable of learning the Q-values corresponding to various online scaling of voltage and frequency settings for the participating system at any state once the model parameters are trained. The computational complexity of the algorithm is expressed as O(MIJ), in which M represents the number of training samples, and I, J, correspond to the input and the obscure units of the autoencoder.

In [67] the authors addressed the joint optimization of task offloading and resource scheduling, formulating it as a mixed-integer non-convex NP-hard problem. To solve this, they introduced a multi-agent deep reinforcement learning approach using the actor-critic method, where agents independently learn optimal offloading policies. Simulation results demonstrated the proposed solution's superior performance compared to benchmark methods. The algorithm complexity is in the order of $O(T \cdot N \cdot A)$, where T is the number of training episodes, N is the number of agents, and A is the size of the action space for each agent.

[68] introduces Mobile Edge Computing (MEC) as a distributed solution, proposing a network management agent that makes offloading decisions from a heterogeneous UE network to MEC servers. Using an Advantage Actor-Critic (A2C) agent, the study demonstrates improved performance over baselines, handling computation, battery, delay, and communication constraints while being scalable, data-efficient, and robust. the overall complexity can be expressed as $O(T \cdot I \cdot N \cdot H)$, where, T represents episodes count, I represents interactions per episode, N represents agents count, and H denotes the hidden units in the neural network.

[69] addresses the computation offloading problem in Mobile Edge Computing (MEC) networks with dynamic weighted tasks. It formulates the joint optimization of offloading decisions and bandwidth allocation as a mixed-integer programming (MIP) problem. The authors propose a Deep Supervised Learning-based Computational Offloading (DSLO) algorithm, which overcomes the limitations of traditional methods by requiring fewer training samples and adapting quickly to new environments. Numerical results demonstrate DSLO's efficiency in achieving high system utility with minimal retraining.

In [70] the authors address task offloading in edge-cloud environments, particularly in Mobile CrowdSourcing (MCS), where resource consumption and service quality often conflict. They propose a Deep Neural Network-based Task Offloading Optimization (DTOO) algorithm to approximate optimal offloading strategies, balancing resource usage and service quality. Additionally, a stack-based offloading strategy with a resource-sorting method reduces task failures. Simulations demonstrate that DTOO improves task completion rates while managing resource conflicts effectively. The algorithm complexity is $O(E \cdot N \cdot W)$, where E represents training epochs count, N represents number of layers in DNN, and W is the total number of weights (parameters) in the DNN.

[71] addresses hardware bottlenecks in smart devices (SDs) executing computation-intensive and delay-sensitive tasks by proposing a deep reinforcement learning-based offloading scheduler (DRL-OS). Using a double dueling deep Q-network (D3QN), the scheduler selects optimal actions—local computing, offloading, or dropping—based on task size, deadline, queue, and battery level. Simulations show DRL-OS reduces battery usage by 54%, drop rates by 42.5%, and average latency to 0.01–0.25 seconds compared to existing methods. the complexity is significantly lower and depends on a single forward pass of the neural network, typically O(Np), where Np is the total number of parameters in the trained network. Deep learning frameworks such as transformer-based real-time scheduling have also been proposed for dynamic MEC resource allocation, showing significant gains in throughput and QoS [72]. Table 7. shows the offloading optimization works based on machine learning.

Table 7: Comparison of offloading methods in MEC environment on machine learning based optimization method

Ref	Optimization Criteria	User Setup	Offloading Method	Task Dependency	Experimental Setup	Algorithm Complexity
[66]	To minimize power consumption	Singleuser	Binary offloading	Independent	Java based simulation	O(MIJ)
[67]	To minimize latency and meet QoS	Multiuser	Binary offloading	Independent	Simulator	O(T·N·A)
[68]	To minimize power consumption and delay	Multiuser	Binary offloading	Independent	Simulator	O(T·I·N·H)
[69]	To minimize resource utility	Multiuser	Binary offloading	Independent	Simulator	N/A
[70]	To minimize task failure rate and achieve QoS	Multiuser	Partial offloading	Independent	Simulator	O(E·N·W)
[71]	To minimize latency, power consumption and task failure rate	Multiuser	Binary offloading	Independent	Simulator	O(Np)

Machine learning approaches show high adaptability to uncertain and dynamic conditions, particularly through reinforcement learning, but none of the coded studies validated them in real-world deployments. Their complexity depends on training overhead rather than closed-form algorithms, meaning scalability is strongly tied to dataset size and model architecture. While promising, they remain at an early stage compared to more mature optimization techniques.

In summary, convex optimization and Lyapunov methods dominate current literature, providing strong theoretical guarantees for energy efficiency and latency minimization, respectively. Heuristics and game theory stand out for their adaptability and practicality in multi-user environments, though their polynomial or cubic complexities limit scalability for very large deployments. Machine learning represents an emerging but under-validated frontier for adaptability under uncertainty, with high training overheads but strong potential. Interpreting complexity metrics further reveals that cubic-time methods (e.g., $O(N^3)$) scale poorly as user numbers grow, whereas lower-order polynomial or logarithmic approaches are more efficient and thus more suitable for large-scale MEC scenarios. These observations reinforce the importance of hybrid strategies that combine the adaptability of ML with the efficiency of convex and heuristic techniques to balance tradeoffs across contexts.

6 Discussions

6.1 Lyapunov optimization

- **Strengths:**
  - Well-suited for dynamic and real-time scenarios without requiring prior statistical data.
  - Efficiently balances multiple objectives such as minimizing latency and energy consumption [48], [49].
  - Adaptable for various user setups and offloading modes, from binary to partial offloading [46-52].
- **Weaknesses:**
  - High computational complexity in scenarios with large task configurations (e.g., $O(|H|^2Tf)$ in [42])
  - Assumes task independence, limiting its application in interdependent task environments.
  - May struggle in environments with highly heterogeneous devices and fluctuating network conditions

6.2 Convex optimization

- **Strengths:**
  - Provides mathematically rigorous solutions with guaranteed convergence for convex problems [53-56].
  - Applicable to energy efficiency and latency minimization in both single-user and multi-user setups.
  - Techniques like sequential programming and variable substitution make non-convex problems tractable [35].
- **Weaknesses:**
  - Requires problem reformulation, which can be computationally intensive.
  - Performance depends on accurate parameter estimation, which may be challenging in real-world scenarios.
  - Not always suitable for highly dynamic systems with frequent state changes.

6.3 Heuristics optimization technique

- **Strengths:**
  - Quick and practical solutions for complex problems where exact methods are computationally expensive [57-62].
  - Easily combined with other approaches, enhancing flexibility and adaptability.
  - Effective for specific goals like cost reduction

[58] or latency minimization [59].

- **Weaknesses:**
  - Lack of theoretical guarantees for optimality.
  - Performance can degrade in large-scale systems with high heterogeneity.
  - Some methods require domain-specific knowledge for parameter tuning and implementation.

6.4 Game theory based optimization

- **Strengths:**
  - Handles multi-user and resource-sharing scenarios effectively [32], [36], [63].
  - Suitable for cooperative and competitive task offloading, ensuring balanced resource allocation [64].
  - Flexible enough to integrate with advanced mechanisms like particle swarm optimization [64].

- **Weaknesses:**
  - Computational complexity increases with the number of participants and resources (e.g., O(Mc·k·Ml) in [32].
  - Assumes rational behavior among participants, which may not always hold in real-world settings.
  - May require extensive simulation to validate results.

6.5 Machine learning based optimization

- **Strengths:**
  - Excellent for dynamic and large-scale systems, leveraging patterns in data for adaptive optimization [67-71].
  - Deep learning models can handle complex scenarios like task dependencies and energy constraints [71].
  - Reinforcement learning methods optimize decisions over time, improving latency and power efficiency [73].

- **Weaknesses:**
  - High computational and training overheads, especially for deep learning models.
  - Performance is sensitive to the quality and volume of training data.
  - Models may require frequent retraining to adapt to new environments, impacting real-time applicability.

Overall, the comparative analysis indicates that Lyapunov and heuristic methods are well suited for latency-sensitive applications, as they provide rapid decision-making and queue stability. In contrast, convex optimization demonstrates the greatest effectiveness in energy-sensitive scenarios, with reported savings of up to 18%. For multi-user and large-scale environments, heuristics and game-theoretic methods show better scalability, although they rely on simplified assumptions. Machine learning approaches offer adaptability to uncertainty and dynamic conditions, but their high training and computational costs remain barriers to deployment. These findings highlight important tradeoffs between scalability, efficiency, and adaptability, and suggest that hybrid approaches may offer the most practical balance for real-world MEC offloading applications.

To strengthen the comparative insights, we systematically coded six representative studies for each of the five major optimization paradigms (Lyapunov, convex, heuristic, game theory, and machine learning). Table 8 summarizes the findings, showing the proportion of simulation-based versus real-world evaluations and the assumptions regarding task dependency. The results highlight that while most approaches rely heavily on simulators and simplifying assumptions, only a limited number incorporate real-world validation. This reinforces the research gap and the need for more practical evaluations under heterogeneous and dynamic MEC conditions.

In summary, Lyapunov and heuristic approaches are preferable for latency-sensitive tasks, convex methods are best suited for energy-sensitive optimization, heuristics and game theory offer better scalability in multi-user environments, while machine learning excels in uncertain and dynamic conditions despite higher training costs; hybrid frameworks that combine these strengths hold the greatest promise for real-world MEC deployments.

Table 8: Comparative summary of coded studies across five optimization paradigms in MEC offloading

Method	Studies (n)	Simulator eval (n, %)	Real-world eval (n, %)	Independent tasks (n, %)	Dependent tasks (n, %)	Example References
Lyapunov	6	6 (100%)	1 (17%) (hybrid)	6 (100%)	0 (0%)	[44]
Convex	6	5 (83%)	1 (17%)	6 (100%)	0 (0%)	[33]
Heuristic	6	4 (67%)	2 (33%)	5 (83%)	1 (17%)	[54], [55]
Game Theory	6	6 (100%)	0 (0%)	1 (17%)	5 (83%)	[30], [34], [60-63]
Machine Learning	6	6 (100%)	0 (0%)	6 (100%)	0 (0%)	[64-69]

To complement the tabular synthesis, Figure 3 presents a radar chart comparing the five optimization paradigms across key dimensions (latency, energy efficiency, scalability, adaptability, and real-world validation), highlighting the distinct strengths and limitations of each approach.

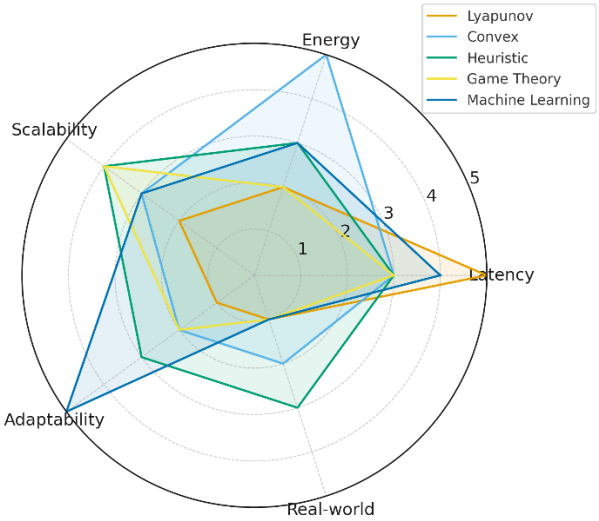

Figure 4: Methodological Comparison Chart

7 Challenges and future work

7.1 Challenges

Our systematic review identified a total of 74 primary studies on computation offloading in Mobile Edge Computing (MEC). To provide a deeper quantitative perspective, we conducted structured coding of 30 representative papers, which are summarized in the accompanying tables. The percentages reported below are derived from this subset of 30 studies, while the broader synthesis across all 74 studies qualitatively supports these same trends.

Ignoring the robust cloud infrastructure: The Majority of computation offloading optimization methods emphasize simplified two-layer architectures. From the 30 papers analyzed in detail, 26 studies (87%) focused on two-layer setups such as SMD–MEC or SMD–Fog, while only 4 papers (13%) addressed more complex three-layer SMD–MEC–Cloud configurations. This reflects a strong bias towards simplified architecture in the field.

Task dependency and device heterogeneity: Most works rely on simplifying assumptions regarding task dependencies and device capabilities. Within the analyzed subset, 24 papers (80%) assumed that tasks were independent, and devices were homogeneous. While this reduces modeling complexity, it limits applicability to heterogeneous real-world environments where tasks are interdependent, and devices vary significantly in resources.

Cost optimization: Balancing energy consumption with performance metrics like latency and task success rates remains a critical challenge. In most of the work, evaluation practices remain dominated by simulations rather than practical deployments. Among the 30 coded papers, 25 studies (83%) validated their methods exclusively in simulated environments, with only a small number incorporating real-world datasets or testbed-based evaluations. This highlights an urgent need for experimental validation to confirm scalability and robustness under realistic conditions.

Computational and scalability issues: Existing approaches such as hierarchical MEC frameworks and metaheuristic algorithms reduce complexity to some extent, but they struggle to adapt efficiently in large-scale and highly dynamic environments. Centralized convex or Lyapunov optimization, while effective in theory, often becomes computationally prohibitive as network size grows. To overcome these issues, future research should prioritize lightweight hybrid methods that combine ML-based prediction with formal optimization, distributed and federated resource allocation frameworks, and collaborative learning between edge and cloud nodes to ensure both efficiency and scalability in real-world deployments.

Technological integration: Limited use of emerging technologies such as 6G, blockchain, quantum computing, and federated learning to address the challenges comprehensively.

7.2 Future directions

Future research should also explore the potential of emerging technologies like 6G networks, blockchain, and quantum computing to enhance MEC systems. 6G networks, with their ultra-low latency and high throughput, can address some of the current network limitations. Blockchain can provide secure and decentralized task management, while quantum computing may offer breakthroughs in solving complex optimization problems. Moreover, the integration of federated learning can help in leveraging distributed data while maintaining user privacy, opening new directions for collaborative offloading decisions.

Principles from adaptive controls such as robust neural adaptive control for uncertain nonlinear systems [74] may inspire hybrid offloading strategies resilient to dynamic MEC conditions.

In addition to the general trends discussed above, task-specific properties such as size, deadlines, and dependencies strongly influence method suitability. Table 9 summarizes these relationships, showing which paradigms align best with different task requirements and where hybrid methods offer added value. Table 9 summarizes these task-specific considerations and their implications for method choice and also the hybrid potential.

Table 9: Task-specific considerations and their influence on method suitability in MEC offloading

Task Property	Key Challenge	Suitable Methods	Remarks / Hybrid Potential
Task size	Large tasks may overload MEC if fully offloaded	Partial offloading, ML prediction	ML can forecast task load, while convex ensures balanced resource use
Deadlines	Stringent deadlines require low-latency decision-making	Lyapunov, heuristics	Hybrid of ML + Lyapunov can adapt quickly under dynamic deadlines
Dependencies	Inter-task constraints (e.g., workflows) complicate scheduling	Heuristics, ML scheduling	Convex/Lyapunov are effective only when tasks are independent
Heterogeneity	Devices vary in resources and mobility	Game theory, ML, hybrid methods	Game theory ensures fairness; ML adapts to device context
QoS / QoE needs	Balancing latency, energy, and reliability	Convex, Lyapunov, hybrid ML-convex	Hybrid approaches can balance optimization guarantees with adaptability

8 Conclusion

This study has provided a comprehensive analysis of computation offloading optimization methods in MEC environments, highlighting the strengths and limitations of various approaches. Techniques ranging from heuristic and convex optimization to machine learning and game theory have been evaluated for their ability to address latency, energy efficiency, and resource allocation challenges.

While significant progress has been made, several challenges, including dynamic resource management, task dependency, and security concerns, continue to hinder the full realization of MEC's potential. The findings emphasize the need for adaptive, secure, and energy-efficient solutions to accommodate the growing complexity of MEC systems. Future advancements, particularly in emerging technologies like 6G, blockchain, and quantum computing, are poised to redefine the landscape of MEC optimization. By addressing these challenges and leveraging these innovations, researchers can unlock the full potential of MEC, enabling seamless and efficient computation offloading for next-generation applications.

References

[1] F. Richter, “Charted: There are more mobile phones than people in the world,” World Economic Forum. Accessed: May 02, 2024. [Online]. Available: https://www.weforum.org/agenda/2023/04/charted-there-are-more-phones-than-people-in-the-world/

[2] J. Hojlo, “Future of Industry Ecosystems: Shared Data and Insights,” IDC. Accessed: May 02, 2024. [Online]. Available: https://blogs.idc.com/2021/01/06/future-of-industry-ecosystems-shared-data-and-insights/

[3] P. Taylor, “Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025,” Statista.

[4] “Cloud Will Be the Centerpiece of New Digital Experiences,” Gartner. Accessed: May 05, 2024. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2021-11-10-gartner-says-cloud-will-be-the-centerpiece-of-new-digital-experiences

[5] M. A. E. Rasool, A. Kumar, and A. Islam, “Dynamic Task Offloading Optimization in Mobile Edge Computing Systems with Time-Varying Workloads Using Improved Particle Swarm Optimization,” Int. J. Adv. Comput. Sci. Appl., vol. 15, no. 4, 2024, doi: http://dx.doi.org/10.14569/IJACSA.2024.0150412.

[6] Y. Li and W. Zhang, “Task-Offloading Strategy of Mobile Edge Computing for WBANs,” 2024.

[7] R. Zhang and W. Shi, “Research on Resource Allocation and Management of Mobile Edge Computing Network,” Informatica, vol. 44, pp. 263–268, 2020.

[8] X. Zhang, “Multi-Objective Resource Allocation in Edge Computing Using Improved Genetic Algorithm with Knowledge-Based Crossover and Segmentation Mutation,” Informatica, vol. 49, no. 25, pp. 213–228, 2025.

[9] P. Mach and Z. Becvar, “Mobile Edge Computing: A Survey on Architecture and Computation Offloading,” IEEE Commun. Surv. Tutorials, vol. 19, no. 3, pp. 1628–1656, 2017, doi: 10.1109/COMST.2017.2682318.

[10] L. Lin, X. Liao, H. Jin, and P. Li, “Computation Offloading Toward Edge Computing,” Proc. IEEE, vol. 107, no. 8, 2019, doi: 10.1109/JPROC.2019.2922285.

[11] T. Zheng, J. Wan, J. Zhang, C. Jiang, and G. Jia, “A Survey of Computation Offloading in Edge Computing,” in 2020 International Conference on Computer, Information and Telecommunication Systems (CITS), IEEE, 2020. doi: 10.1109/CITS49457.2020.9232457.

[12] A. Shakarami, A. Shahidinejad, and M. Ghobaei-Arani, “A review on the computation offloading

approaches in mobile edge computing: A game-theoretic perspective," Softw. Pract. Exp., vol. 50, no. 9, pp. 1719–1759, 2020, doi: https://doi.org/10.1002/spe.2839.

[13] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective," Comput. Networks, vol. 182, 2020, doi: https://doi.org/10.1016/j.comnet.2020.107496.

[14] Q.-H. Nguyen and F. Dressler, "A smartphone perspective on computation offloading—A survey," Comput. Commun., vol. 159, pp. 133–154, 2020, doi: https://doi.org/10.1016/j.comcom.2020.05.001.

[15] A. Agarwal, V. Gupta, and P. Singh, "Game-theoretic approaches for computation offloading in mobile edge computing: A survey," IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 2204–2232, 2018.

[16] S. Thotla and S. Mahajani, "Energy-efficient heuristic methods for computation offloading in mobile edge computing: A comprehensive review," Future Generation Computer Systems, vol. 95, pp. 454–468, 2019.

[17] M. Hasan, M. Z. Chowdhury, and Y. M. Jang, "Computation offloading and resource allocation in mobile edge computing: A survey of convex and Lyapunov optimization techniques," IEEE Access, vol. 8, pp. 197–214, 2020.

[18] A. Singh, R. Sharma, and K. S. Patel, "Computation offloading in IoT using mobile edge computing: A survey," Journal of Network and Computer Applications, vol. 171, p. 102785, 2021.

[19] L. Zheng and L. Cai, "A Distributed Demand Response Control Strategy Using Lyapunov Optimization," IEEE Trans. Smart Grid, vol. 5, no. 4, pp. 2075–2083, 2014. doi: 10.1109/TSG.2014.2313347.

[20] Y. Mao, J. Zhang, and K. B. Letaief, "A Lyapunov Optimization Approach for Green Cellular Networks With Hybrid Energy Supplies," IEEE J. Sel. Areas Commun., vol. 33, no. 12, pp. 2463–2477, 2015, doi: 10.1109/JSAC.2015.2481209.

[21] A. Ben-Tal and A. Nemirovski, LECTURES ON MODERN CONVEX OPTIMIZATION ANALYSIS, ALGORITHMS, AND ENGINEERING APPLICATIONS. Society for Industrial and Applied Mathematics Philadelphia, 2001. [Online]. Available: https://epubs.siam.org/doi/pdf/10.1137/1.9780898718829.fm

[22] S. P. Boyd and L. Vandenberghi, Convex Optimization. Cambridge University Press, 2004.

[23] K. Y. Lee and M. A. El-Sharkawi, Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems. WILEY INTERSCIENCE, 2008.

[24] R. B. Myerson, Game Theory Analysis of Conflict. Harvawrd University Press, 1997. doi: 10.1109/TVT.2017.2764002.

[25] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, AAAI Press, 2018, pp. 3207–3214. doi: 10.1609/aaai.v32i1.11694.

[26] D. P. Bertsekas, Reinforcement Learning and Optimal Control. Athena Scientific, Belmont, MA, 2019.

[27] P. Paymard, S. Rezvani, and N. Mokari, "Joint task scheduling and uplink/downlink radio resource allocation in PD-NOMA based mobile edge computing networks," Phys. Commun., vol. 32, pp. 160–171, 2019, doi: https://doi.org/10.1016/j.phycom.2018.11.00

[28] H. Yu, Q. Wang, and S. Guo, "Energy-Efficient Task Offloading and Resource Scheduling for Mobile Edge Computing," in 2018 IEEE International Conference on Networking, Architecture and Storage (NAS), 2018. doi: 10.1109/NAS.2018.8515731.

[29] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices," IEEE J. Sel. Areas Commun., vol. 34, no. 12, pp. 3590–3605, 2016, doi: 10.1109/JSAC.2016.2611964.

[30] P. Liu, G. Xu, K. Yang, K. Wang, and X. Meng, "Jointly Optimized Energy-Minimal Resource Allocation in Cache-Enhanced Mobile Edge Computing Systems," IEEE Access, vol. 7, pp. 3336–3347, 2019, doi: 10.1109/ACCESS.2018.2889815.

[31] Y. Yang, Y. Ma, W. Xiang, X. Gu, and H. Zhao, "Joint Optimization of Energy Consumption and Packet Scheduling for Mobile Edge Computing in Cyber-Physical Networks," IEEE Access, vol. 6, pp. 15576–15586, 2018, doi: 10.1109/ACCESS.2018.2810115.

[32] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint Computation Offloading and Resource Allocation Optimization in Heterogeneous Networks with Mobile Edge Computing," IEEE Access, vol. 6, pp. 19324–19337, 2018, doi: 10.1109/ACCESS.2018.2819690.

[33] X. Chen, H. Zhang, C. Wu, S. Mao, and Y. Ji, "Optimized Computation Offloading Performance in Virtual Edge Computing Systems Via Deep Reinforcement Learning," IEEE Internet Things J., vol. 6, no. 3, pp. 4005–4018, 2019, doi: 10.1109/JIOT.2018.2876279.

[34] Z. Tan, F. R. Yu, X. Li, H. Ji, and V. C. M. Leung, "Virtual Resource Allocation for Heterogeneous Services in Full Duplex-Enabled SCNs With Mobile Edge Computing and Caching," IEEE

Trans. Veh. Technol., vol. 67, no. 2, pp. 1794–1808, 2017.

[35] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-Efficient Dynamic Computation Offloading and Cooperative Task Scheduling in Mobile Cloud Computing," IEEE Trans. Mob. Comput., vol. 18, no. 2, pp. 319–333, 2019, doi: 10.1109/TMC.2018.2831230.

[36] Q.-V. Pham, H. T. Nguyen, Z. Han, and W.-J. Hwang, "Coalitional Games for Computation Offloading in NOMA-Enabled Multi-Access Edge Computing," EEE Trans. Veh. Technol., vol. 60, no. 2, pp. 1982–1993, 2020, doi: 10.1109/TVT.2019.2956224.

[37] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in 2018 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2018. doi: 10.1109/WCNC.2018.8377343.

[38] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," J. VLSI signal Process. Syst. signal, image video Technol., vol. 13, pp. 203–221, 1996, doi: https://doi.org/10.1007/BF01130406.

[39] Z. Song, Y. Liu, and X. Sun, "Joint Radio and Computational Resource Allocation for NOMA-Based Mobile Edge Computing in Heterogeneous Networks," IEEE Commun. Lett., vol. 22, no. 12, pp. 2559–2562, 2018. doi: 10.1109/LCOMM.2018.2875984.

[40] X. Long, J. Wu, and L. Chen, "Energy-Efficient Offloading in Mobile Edge Computing with Edge-Cloud Collaboration," in International Conference on Algorithms and Architectures for Parallel Processing, 2018, pp. 460–475. doi: https://doi.org/10.1007/978-3-030-05057-3_35.

[41] Z. Tong, X. Deng, F. Ye, S. Basodi, X. Xiao, and Y. Pan, "Adaptive computation offloading and resource allocation strategy in a mobile edge computing environment," Inf. Sci. (Ny)., vol. 537, pp. 116–131, 2020. doi: https://doi.org/10.1016/j.ins.2020.05.057

[42] Y. Cui, D. Zhang, T. Zhang, L. Chen, M. Piao, and H. Zhu, "Novel method of mobile edge computation offloading based on evolutionary game strategy for IoT devices," AEU - Int. J. Electron. Commun., vol. 118, 2020, doi: https://doi.org/10.1016/j.aeue.2020.153134.

[43] Z. S. C. Yi, J. Cai, "A Multi-User Mobile Computation Offloading and Transmission Scheduling Mechanism for Delay-Sensitive Applications," IEEE Trans. Mob. Comput., vol. 19, no. 1, pp. 29–43, 2020, doi: 10.1109/TMC.2019.2891736.

[44] Z. Zhu, J. Peng, X. Gu, H. Li, K. Liu, and Z. Zhou, "Fair Resource Allocation for System Throughput Maximization in Mobile Edge Computing," IEEE Access, vol. 6, pp. 5332–5340, 2018, doi: 10.1109/ACCESS.2018.2790963.

[45] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, Spain, 2018, 2018, pp. 1–6. doi: 10.1109/WCNC.2018.837734.

[46] U. Rahul et al., "Dynamic service migration and workload scheduling in edge-clouds," Perform. Eval., vol. 91, pp. 205–228, 2015, doi: 10.1016/J.PEVA.2015.06.013.

[47] M. Guo, W. Wang, X. Huang, Y. Chen, L. Zhang, and L. Chen, "Lyapunov-Based Partial Computation Offloading for Multiple Mobile Devices Enabled by Harvested Energy in MEC," IEEE Internet Things J., vol. 9, no. 11, pp. 9025–9038, 2022, doi: 10.1109/JIOT.2021.3108381.

[48] K. Katsalis, T. G. Papaioannou, N. Nikaein, and L. Tassiulas, "SLA-Driven VM Scheduling in Mobile Edge Computing," in IEEE 9th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, 2016, pp. 750–757. doi: 10.1109/CLOUD.2016.0104.

[49] X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, and G. B. Giannakis, "Optimal schedule of mobile edge computing for internet of things using partial information," IEEE J. Sel. Areas Commun., vol. 35, no. 11, pp. 2606–2615, 2017. doi: 10.1109/JSAC.2017.2760186.

[50] S. Xia, Z. Yao, Y. Li, and S. Mao, "Online distributed offloading and computing resource management with energy harvesting for heterogeneous mec-enabled iot," IEEE Trans. Wirel. Commun., vol. 20, no. 10, pp. 6743–6757, 2021, doi: 10.1109/TWC.2021.3076201.

[51] C. Li, J. Tang, and Y. Luo, "Dynamic multi-user computation offloading for wireless powered mobile edge computing," J. Netw. Comput. Appl., vol. 131, pp. 1–15, 2019, doi: https://doi.org/10.1016/j.jnca.2019.01.020.

[52] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, 2017. doi: 10.1109/WCNC.2017.7925615.

[53] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobileedge computation offloading," IEEE Trans. Wirel. Commun., vol. 16, no. 3, pp. 1397–1411, 2017. doi: 10.1109/TWC.2016.2633522.

[54] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," IEEE Trans. Wirel. Commun., vol. 18, no. 1, pp. 695–708, 2019, doi: 10.1109/TWC.2018.2885266.

[55] M. Salmani and T. N. Davidson, “Uplink resource allocation for multiple access computational offloading,” Signal Processing, vol. 168, 202AD. doi: https://doi.org/10.1016/j.sigpro.2019.107322

[56] C. Li, J. Tang, H. Tang, and Y. Luo, “Collaborative cache allocation and task scheduling for data-intensive applications in edge computing environment,” Futur. Gener. Comput. Syst., vol. 95, pp. 249–264, 2019. doi: https://doi.org/10.1016/j.future.2019.01.007.

[57] C. Li, J. Bai, Y. Chen, and Y. Luo, “Resource and replica management strategy for optimizing financial cost and user experience in edge cloud computing system,” Inf. Sci. (Ny)., vol. 516, pp. 33–55, 2020, doi: https://doi.org/10.1016/j.ins.2019.12.049.

[58] M. Ghobaei-Arani, A. Souri, F. Safara, and M. Norouzi, “An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing,” Emerg. Telecommun. Technol., vol. 31, no. 2, 2020. doi: https://doi.org/10.1002/ett.3770.

[59] L. Yang, C. Zhong, Q. Yang, W. Zou, and A. Fathalla, “Task offloading for directed acyclic graph applications based on edge computing in industrial internet,” Inf. Sci. (Ny)., vol. 540, pp. 51–68, 2020. doi: https://doi.org/10.1016/j.ins.2020.06.001.

[60] W. Bai and Y. Wang, “Jointly Optimize Partial Computation Offloading and Resource Allocation in Cloud-Fog Cooperative Networks,” pp. 1–20, 2023

[61] D. Lv, P. Wang, Q. Wang, Y. Ding, Z. Han, and Y. Zhang, “Task Offloading and Resource Optimization Based on Predictive Decision Making in a VIoT System,” pp. 1–21, 2024.

[62] J. Zhou, X. Zhang, and W. Wang, “Joint resource allocation and user association for heterogeneous services in multi-access edge computing networks,” IEEE Access, vol. 7, pp. 12272–12282, 2019. doi: 10.1109/ACCESS.2019.2892466.

[63] S. Wang, Z. Hu, Y. Deng, and L. Hu, “Game-Theory-Based Task Offloading and Resource Scheduling in Cloud-Edge Collaborative Systems,” 2022.

[64] L. Wang, W. Zhou, H. Xu, L. Li, and L. Cai, “Research on task offloading optimization strategies for vehicular networks based on game theory and deep reinforcement learning,” no. October, pp. 1–17, 2023. doi: 10.3389/fphy.2023.1292702.

[65] K. Zhang, J. Yang, and Z. Lin, “Computation Offloading and Resource Allocation Based on Game Theory in Symmetric MEC-Enabled Vehicular Networks,” 2023.

[66] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, and P. Li, “Energy-efficient scheduling for real-time systems based on deep q-learning model,” IEEE Trans. Sustain. Comput., vol. 4, no. 1, pp. 132–141, 2019, doi: 10.1109/TSUSC.2017.2743704.

[67] R. Garaali, C. Chaieb, W. Ajib, and M. Afif, “Learning-Based Task Offloading for Mobile Edge Computing,” in ICC 2022 - IEEE International Conference on Communications, 2022. doi: 10.1109/ICC45855.2022.9838831.

[68] C. Silva, N. Magaia, and A. Grilo, “Task Offloading Optimization in Mobile Edge Computing based on Deep Reinforcement Learning,” in MSWiM ’23: Proceedings of the Int’l ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems, 2023, pp. 109–118. doi: https://doi.org/10.1145/3616388.3617539.

[69] S. Yang, G. Lee, and L. Huang, “Deep Learning-Based Dynamic Computation Task Offloading for Mobile Edge Computing Networks,” pp. 1–18, 2022.

[70] L. Meng, Y. Wang, H. Wang, X. Tong, Z. Sun, and Z. Cai, “Task offloading optimization mechanism based on deep neural network in edge - cloud environment,” J. Cloud Comput., 2023. doi: 10.1186/s13677-023-00450-6.

[71] D. Lim, W. Lee, W. T. Kim, and I. Joe, “DRL-OS: A Deep Reinforcement Learning-Based Offloading Scheduler in Mobile Edge Computing,” Sensors, vol. 22, no. 23, 2022. doi: 10.3390/s22239212.

[72] J. Chao and M. Jiao, “Network Spectrum Resource Allocation and Optimization Based on Deep Learning and TRDM,” Informatica, vol. 49, no. 13, pp. 105–122, 2025

[73] D. Lim, W. Lee, W. T. Kim, and I. Joe, “DRL-OS: A Deep Reinforcement Learning-Based Offloading Scheduler in Mobile Edge Computing,” Sensors, vol. 22, no. 23, 2022, doi: 10.3390/s22239212.

[74] L. Ma, Y. Pan, H. Yu, and H. He, “Robust neural adaptive control for a class of uncertain nonlinear complex dynamical multivariable systems,” Neurocomputing, vol. 138, pp. 200–208, 2014. doi: 10.1016/j.neucom.2014.01.031