Task Offloading of Edge Computing Based on Reinforcement Learning: A Survey

Pu Yangyizhen, Xie Yijie, Zheng Tongzhou, Zhou Xufei
(in alphabetical order of pinyin)

***Abstract*—Edge computing integrates multiple core capabilities on the network edge close to the data source. However, the growth of device data processing and task requirements has brought challenges, leading to the emergence of task offloading. This paper focuses on task offloading in edge computing based on reinforcement learning and elaborates on its research status. In terms of the foundation of reinforcement learning, it expounds the principles and advantages of deep Q-network (DQN) and its improved algorithms such as dueling double deep Q-network (D3QN). In the application aspect, it covers the applications of classical reinforcement learning (such as Q-learning) and deep reinforcement learning (such as D3QN, multi-agent reinforcement learning MARL) in task offloading, including key elements such as the definition of state and action and reward mechanisms. For reinforcement learning methods, it introduces multiple optimization methods such as DOQO and TRL framework and strategies for compressing action and state spaces. It also analyzes the challenges such as scenario design and algorithm complexity and looks ahead to future development directions such as efficient algorithm design and multi-objective optimization, providing a comprehensive reference for the research and application of edge computing task offloading technology.**

***Index Terms*—Edge Computing, Task Offloading, Reinforcement Learning, Multi-Agent Reinforcement Learning.**

I. Introduction

EDGE computing refers to a distributed and open platform that integrates the core capabilities of networking, computing, storage, and applications on the network edge side close to the data source. It provides edge intelligence services nearby to meet the key requirements of industry digitalization in aspects such as agile connection, real-time services, data optimization, application intelligence, security, and privacy protection [1]. Traditional cloud computing generally centralizes data in data centers for processing, while edge computing assigns some computing tasks to the network edge for processing, handling data where it is generated.

The importance of edge computing is becoming increasingly prominent. Taking the booming autonomous driving in recent years as an example, autonomous driving requires extremely high real-time performance. If traditional cloud computing is used, the delay caused by vehicles transmitting data to the data center for processing and then getting the results back is unacceptable and may even lead to traffic accidents. However, edge computing enables vehicles to process data locally and make timely decisions, ensuring real-time performance.

Meanwhile, edge computing has also become important in terms of data privacy protection. For instance, many sensitive data (such as medical and health data, financial transaction data, etc.) are not suitable for being transmitted to remote cloud for processing. Edge computing allows data processing and analysis to be carried out on local devices or edge servers, reducing the risk of data leakage.

Although edge computing is gradually becoming a key technology to support various emerging applications and business models, with the development of mobile devices and the Internet of Things, the amount of data that devices need to handle and computing tasks are becoming increasingly complex. To further improve the processing efficiency, task offloading has emerged.

Task offloading means transferring the computing tasks of a device to other devices or servers with stronger computing capabilities for processing. It is common to offload the tasks of mobile devices to edge servers or the cloud [2].

However, in actual dynamic environments, task offloading faces numerous challenges:

- **Latency Challenge**: In dynamic environments, the network state can change at any time. Situations such as network congestion and unstable signals may lead to a significant increase in data transmission latency. For example, in the Internet of Vehicles scenario, vehicles are in a high-speed moving state and network connections are constantly switching. During the process of offloading tasks from vehicles to edge servers or the cloud, it is difficult to predict and control the data transmission latency [3].
- **Energy Consumption Challenge**: Although offloading tasks to external computing resources can utilize stronger computing capabilities, the energy consumption issue during the task offloading process also needs to be considered. Packaging and transmitting data as well as interacting with external resources all consume energy. Especially for mobile devices with limited battery power, an unreasonable task offloading strategy may cause a sharp increase in device energy consumption and shorten the device's battery life [4].
- **Device Heterogeneity Challenge**: The development of mobile devices and the Internet of Things has led to an increasingly diverse range of devices, with huge differences in hardware architectures, computing capabilities, storage capacities, and so on. This heterogeneity has caused two main problems.
  - On the one hand, the software and hardware environments of different devices are different. If the offloaded tasks are to run smoothly on the target devices, complex compatibility adaptation work is

required.
- On the other hand, the performance and computing capabilities of different devices also vary. If the same tasks are assigned to each device, it will lead to a decline in performance.

Therefore, tasks need to be reasonably allocated according to the actual situation of the devices.

Faced with a series of complex challenges such as latency, energy consumption, and device heterogeneity that task offloading encounters in dynamic environments, traditional methods based on rules or static optimization are gradually proving to be inadequate. These traditional methods often struggle to cope with the dynamic changes of the environment and complex system constraints, and thus cannot achieve optimal performance.

Against this background, reinforcement learning, as a powerful machine learning technique, has provided highly promising ideas and methods for solving the problem of task offloading.

The core of reinforcement learning lies in the fact that an agent interacts with the environment, continuously tries different actions, and learns the optimal strategy based on the obtained reward feedback, so as to maximize the cumulative reward in the long term [5]. This learning mode is inherently suitable for the dynamically changing task offloading scenarios because it can dynamically adjust task offloading decisions according to the real-time environmental states (such as network conditions, device loads, etc.), achieving self-adaptation to complex environments.

For example, when dealing with the latency issue, the reinforcement learning algorithm can, through continuous trial-and-error learning, choose to offload tasks to local devices, edge servers, or the cloud according to the real-time latency situation of the current network, so as to ensure that the overall task execution latency is minimized [6].

Through continuous learning and optimization, reinforcement learning can help agents find near-optimal task offloading strategies, providing a powerful technical support for solving the difficulties faced by task offloading in dynamic environments. It is expected to become a key tool for promoting the development of edge computing task offloading technology.

II. Research Status of Reinforcement Learning in Task Offloading

A. The Foundation of Reinforcement Learning

The theory of reinforcement learning provides a normative account, deeply rooted in psychological and neuroscientific perspectives on animal behaviour, of how agents may optimize their control of an environment. To use reinforcement learning successfully in situations approaching real-world complexity, however, agents are confronted with a difficult task: they must derive efficient representations of the environment from high-dimensional sensory inputs, and use these to generalize past experience to new situations.While reinforcement learning agents have achieved some successes in a variety of domains, their applicability has previously been limited to domains in which useful features can be handcrafted, or to domains with fully observed, low-dimensional state spaces. To address this issue, researchers proposed the Deep Q-Network (DQN), which combines deep learning with Q-learning to enable effective decision-making in complex environments [7].

DQN uses one particularly successful architecture, the deep convolutional network, which uses hierarchical layers of tiled convolutional filters to mimic the effects of receptive fields—inspired by Hubel and Wiesel's seminal work on feedforward processing in early visual cortex—thereby exploiting the local spatial correlations present in images, and building in robustness to natural transformations such as changes of viewpoint or scale.

Reinforcement learning is known to be unstable or even to diverge when a nonlinear function approximator, such as a neural network, is used to represent the action-value (Q) function. This instability has several causes, including the correlations present in the sequence of observations, the fact that small updates to Q may significantly change the policy and therefore change the data distribution, and the correlations between the action-values (Q) and the target values. There is a novel variant of Q-learning that employs two key ideas:

- **Experience Replay**: This biologically inspired mechanism randomizes the data, thereby removing correlations in the observation sequence and smoothing over changes in the data distribution. Furthermore, this mechanism stores the agent's experiences (state, action, reward, next state) in a replay buffer. During training, small batches of samples are randomly drawn from this buffer to update the network, breaking the correlations between consecutive data points and improving learning efficiency.
- **Iterative Update**: This method adjusts the action-values (Q) towards target values that are only periodically updated, thereby reducing correlations with the target.

To stabilize the training process, DQN employs the concept of a target network. The weights of the target network are updated periodically (e.g., every few steps) to reduce fluctuations in the Q-values. By using a target network, DQN maintains a relatively stable target during updates, which enhances learning stability and convergence speed.The training process of DQN includes the following steps:

1) Action Selection: The agent selects an action based on the current state, typically using an ϵ-greedy strategy, where it randomly selects an action with a certain probability (exploration) and selects the action with the highest Q-value with a certain probability (exploitation).

2) Action Execution: The agent executes the selected action in the environment and observes the next state and reward.

3) Experience Storage: The current experience state,action,reward,nextstate is stored in the replay buffer.

4) Q-value Update: A small batch of samples is randomly drawn from the replay buffer, and the Q-values are updated using the Bellman equation. Specifically, DQN updates the network weights by minimizing the mean squared error of the Q-values.

The success of DQN demonstrates the potential of combining deep learning with reinforcement learning, laying the foundation for subsequent research. This method not only achieved

significant results in gaming but also inspired applications in other fields, such as robotics and autonomous driving.

Dueling Double Deep Q-Network (D3QN) is an improvement over the Deep Q-Network (DQN) designed to enhance the learning efficiency and stability of reinforcement learning in complex environments. D3QN combines the advantages of Double Q-learning and the dueling network architecture, primarily consisting of the following key points:

- **Double Q-Learning**:D3QN introduces the concept of Double Q-learning to address the overestimation problem of Q-values in traditional Q-learning. By using two independent Q-networks (the main network and the target network), D3QN selects actions using one network and evaluates the value of those actions using the other network, thereby reducing bias in the Q-values.
- **Dueling Network Architecture**:D3QN employs a dueling network architecture that decomposes the Q-value function into two components: the state value function (V) and the advantage function (A) [8]. This decomposition allows the network to better capture the relative value of different actions. Specifically, the Q-value can be expressed as:

$$Q(s,a) = V(s) + A(s,a) \tag{1}$$

  By using this approach, D3QN provides a more stable learning signal, especially in situations where states are similar but actions differ.
- **Experience Replay and Target Network**:D3QN continues to utilize the experience replay mechanism and target network from DQN to further enhance learning stability [9].Experience replay breaks the correlations between consecutive data points by randomly sampling from a replay buffer, while the target network is updated periodically to reduce fluctuations during training.
- **Performance Improvement**:Experiments have shown that D3QN performs exceptionally well in various reinforcement learning tasks, particularly in environments that require handling complex state and action spaces. By combining Double Q-learning and the dueling network architecture, D3QN converges to optimal policies more quickly and improves overall performance.

B. Applications of Classical Reinforcement Learning

Q-learning The breakthrough in Machine Learning (ML) techniques and the popularity of the Internet of Things (IoT) has increased interest in applying Artificial Intelligence (AI) techniques to the new paradigm of Edge Computing. One of the challenges in edge computing architectures is the optimal distribution of the generated tasks between the devices in each layer (i.e., cloud-fog-edge) [10].

To make efficient use of devices in edge computing architecture, it is necessary to use a computational offloading framework that optimizes the allocation of tasks to computing devices in the best possible way. The procedure for deciding the best allocation of tasks to devices is called "task assignment problem"and is a combinatorial optimization problem defined as the process of determining where the computation of each task is performed in order to minimize certain parameters, such as the aforementioned latency and energy consumption, an important topic for an eco-friendly future [11] and a key component of the green computing philosophy.

The task assignment problem (TAP) can be formulated as a Generalized Assignment Problem (GAP) [12], which is an NP-hard. Among the traditional methods for solving GAP, new methods based on dynamic programming and machine learning techniques have emerged, such as reinforcement learning (RL) and neural network reinforcement learning (Deep RL). Reinforcement learning optimizes task offloading through the following key aspects:

- **Definition of State and Action**:Researchers define a state space that includes various factors such as the device's computing capability, network latency, task size, and complexity. This state information helps the agent understand the characteristics of the current environment, enabling it to make more informed decisions. The action space includes different offloading strategies, such as choosing to process tasks locally, offloading tasks to edge servers, or selecting different edge servers for processing. By defining multiple possible actions, the agent can explore various offloading strategies.
- **Reward Mechanism**:Researchers design a reward mechanism aimed at encouraging the agent to select offloading strategies that minimize latency and energy consumption. After executing each action, the agent receives corresponding rewards or penalties based on task completion time, energy consumption, and other performance metrics. This feedback mechanism prompts the agent to continuously adjust its strategy to optimize overall performance.
- **Multi-Layer Guided Mechanism**:The multi-layer guided mechanism proposed allows the agent to learn at different decision-making levels. Each layer's guiding strategy targets specific task features and environmental states, helping the agent learn and make decisions more effectively in complex task offloading scenarios. This hierarchical structure enables the agent to converge to the optimal strategy more quickly.
- **Reinforcement Learning Algorithm**:Specific reinforcement learning algorithms (such as Q-learning or Deep Q-Networks) can train the agent. Through interaction with the environment, the agent continuously updates its strategy to adapt to the dynamically changing edge computing environment.

All innovation comes with challenges, and IoT and mobile technologies are no exception. One of them is the latency problem, some mobile applications, such as autonomous driving or virtual reality, need to meet a maximum delay requirement, so in some cases it is not possible to send the data to the cloud. Reinforcement learning has the following advantages in addressing these issues:

1) Adaptability: Reinforcement learning can dynamically adjust its strategy based on changes in the environment. In edge computing, network conditions, device loads, and user demands may change frequently, and reinforcement learning

can learn and adapt in real-time to optimize task offloading and resource allocation.

2) Efficient Resource Management: Edge computing often faces challenges with limited resources. Reinforcement learning can learn optimal resource allocation strategies to maximize resource utilization, reduce latency, and minimize energy consumption. For example, agents can learn how to select the most suitable computing node to process tasks under different network conditions.

3) Online Learning Capability: Reinforcement learning can learn during operation, meaning it can continuously update and optimize its strategy in edge computing environments without needing offline training. This online learning capability allows the system to quickly adapt to new tasks and changes in the environment.

4) Reduced Communication Overhead: In edge computing, transmitting data to the cloud for processing can lead to high latency and bandwidth consumption. By using reinforcement learning on edge devices, agents can decide when and how to offload tasks, thereby reducing unnecessary communication overhead.

5) Support for Multi-task Processing: There may be multiple tasks occurring simultaneously in edge computing environments. Reinforcement learning can learn how to effectively schedule and allocate resources among multiple tasks to achieve higher system throughput and efficiency.

Through these methods, reinforcement learning achieves an adaptive and intelligent decision-making process in task offloading, allowing edge computing systems to handle tasks more efficiently. In the future, people may explore alternatives of knowledge transfer and federated learning to develop distributed decision systems in 5G networks and Computing Continuum paradigm [13]. Another promising future research direction is to apply neural networks to improve the performance of Q-Learning, thus overcoming some limitations such as a low number of states and discrete variables.

C. Applications of Deep Reinforcement Learning

In the research on task offloading, enhanced reinforcement learning methods have also been continuously developed and innovated. Here, we mainly introduce two methods. One is the Double Dueling DQN (D3QN), which demonstrates unique advantages in solving offloading problems in highly dynamic environments. The other is Multi-Agent Reinforcement Learning (MARL), which plays an important role in edge collaboration scenarios.

Edge computing networks face difficulties such as limited device resources and dynamic environmental changes. Zhang et al. [14] modeled the online offloading problem as a Markov Decision Process (MDP) and, on this basis, proposed the C - D3QN algorithm [14]. This algorithm combines the D3QN algorithm with a clustering algorithm and effectively improves the decision-making performance in highly dynamic environments by introducing a double dueling mechanism.

In terms of algorithm design, the centralized agent collects information on edge servers, mobile terminal devices, and channel states to form the state space. The action space covers the task execution decisions of terminal devices, and the reward function combines incentives such as task completion latency, terminal energy consumption, and task success rate to guide the agent's decisions.

When it comes to exploration and action selection, the agent adopts the ε-greed method. In the early stage, it randomly explores with a high probability, and in the later stage, it selects the action with the maximum Q value. The C - D3QN model introduces a fixed target mechanism, initializing the online learning and target networks. The online learning network updates parameters in real time, and the target network is periodically updated to the parameter values of the online learning network to stabilize training and avoid overestimation of Q values. Meanwhile, the dueling mechanism changes the network output from a single Q value to a value function V value and an advantage function A value, enabling a more accurate estimation of Q values and enhancing the adaptability to highly dynamic environments.

Experiments show that the C - D3QN algorithm has obvious advantages over DQN in complex dynamic scenarios. It has a faster convergence speed and can quickly adapt to environmental changes for decision-making. In terms of task completion latency, it selects appropriate execution methods according to tasks and resources to reduce latency. Regarding terminal energy consumption, it makes reasonable offloading decisions to avoid excessive energy consumption and extend the battery life.

As the scale of edge computing expands, the collaboration among edge nodes becomes increasingly important. Multi-Agent Reinforcement Learning (MARL) can better cope with complex edge collaboration scenarios through the interaction and collaboration among multiple agents.

In edge computing task offloading, different edge servers or devices can be regarded as different agents. They work together to complete the offloading tasks while also taking into account factors such as the communication costs and differences in computing capabilities among different devices and servers [15]. The MARL approach allows each agent to make decisions based on its own observations and local information, and at the same time communicate and collaborate with other agents to gradually achieve the globally optimal collaboration strategy.

However, MARL also faces challenges. Conflicts of interest among agents may lead to difficulties in collaboration, and an increase in the number of agents causes the state and action spaces to grow exponentially, resulting in the "curse of dimensionality" and increasing the computational complexity and convergence difficulty. To address these challenges, researchers have proposed various solutions. A reasonable reward mechanism can adjust the reward function, rewarding agents that actively collaborate and punishing selfish behaviors, prompting them to pursue their own interests while achieving the goals of the system [16]. The hierarchical MARL architecture divides agents into different layers, with higher-level agents coordinating lower-level ones, reducing complexity and improving collaboration efficiency [17].

In practical applications, MARL algorithms can support edge devices to coordinate the allocation of task shares and

resource utilization in real time. They can also assist vehicles and Roadside Units (RSUs) in sharing traffic information and facilitating task offloading. In the future, MARL algorithms are also expected to play a significant role in the Internet of Things.

In conclusion, the applications of Double Dueling DQN and Multi-Agent Reinforcement Learning in the field of edge computing task offloading provide powerful means to solve the problems in highly dynamic environments and edge collaboration. However, they also face numerous challenges at the same time. Future research needs to further optimize the algorithms to adapt to the continuously developing demands of edge computing and promote the application and development of edge computing technology in a broader range of fields.

III. Optimization and Innovation of Reinforcement Learning Methods

A. Method Optimization

The task offloading technology plays a crucial role in edge computing systems, directly impacting both system efficiency and user experience. However, in real-world applications, task offloading faces several challenges, particularly in the context of delay-sensitive tasks. In areas such as autonomous driving and healthcare, task deadlines are stringent, and any delays may result in serious consequences. Additionally, the heterogeneity of devices and tasks introduces significant complexity to task offloading. Differences in resource capacities, task complexity, and data characteristics complicate the formulation of a unified offloading strategy. Achieving a balance between generic policies and individualized requirements remains a significant challenge, as centralized policy training, despite its low overhead, lacks specificity, while individualized policies incur high training costs. The cold-start problem further exacerbates these issues; newly introduced devices may not be covered by existing policies, requiring retraining and adding to system overhead.

To address these challenges, several approaches have been proposed. The DOQO method, as introduced by Chen et al., enhances real-time performance and optimizes resource utilization by dynamically adjusting offloading decisions [18]. The Transfer Reinforcement Learning (TRL) framework, developed by Shuai et al., combines transfer learning and domain adaptation to reduce training costs and improve model adaptability [19]. The C-D3QN algorithm, proposed by Zhang et al., uses K-means clustering and deep reinforcement learning to optimize task offloading, significantly enhancing exploration efficiency [20]. Robles-Enciso et al. introduced the Multi-Layer Reinforcement Learning (ML-RL) system, which improves task offloading efficiency through cooperation among the edge, fog, and cloud layers [21]. Wang et al. proposed MRLCO, a meta-weighted learning-based system, which rapidly adapts to dynamic environments while maintaining high sample efficiency and stable performance. [22]

1) DOQO: DOQO is a real-time, sense-dependent task offloading method that primarily addresses the task offloading problem in dynamic environments. The method optimizes the task offloading process by modeling mobile applications as directed acyclic graphs (DAGs) to capture inter-task dependencies and parallelism.DOQO employs a deep Q-network (DQN) algorithm to evaluate the Q-value of the offloading decision and dynamically adjust the offloading strategy. A distinguishing feature of DOQO is its approach to task prioritization, which eschews preset priorities in favor of a flexible scheduling mechanism that responds to real-time changes in the environment. This flexibility enables the system to swiftly generate offloading plans and make timely adjustments, a capability that traditional methods often lack.

However, DOQO presents several challenges. The uncertainty in task scheduling and execution location leads to an expansive solution space, increasing computational complexity. Conflicting priorities may also hinder the efficient execution of parallel tasks. Furthermore, DOQO's disregard for wireless channel fading effects may limit its effectiveness in real-world mobile environments. Despite these limitations, DOQO remains a viable method, and further optimization is required, particularly in its adaptability to complex environments. [18]

2) TRL Framework: The TRL framework proposes a task offloading method that combines migration learning with domain adaptive mechanisms. The objective of this combination is to reduce training overhead and improve model adaptation. The framework is composed of two modules: a data alignment module and a reinforcement learning module. The data alignment module maps data from disparate devices to the reproducing kernel Hilbert space (RKHS) through the domain adaptation mechanism, thereby reducing data distribution differences and enhancing the sharing capability among devices. The reinforcement learning module employs an Actor-Critic architecture, enabling devices to make globally optimal decisions under limited observation conditions and optimize strategies through feedback. The incorporation of Migration Learning has been demonstrated to expedite the convergence of Deep Reinforcement Learning (DRL) models, thereby facilitating the rapid adaptation of devices to novel environments and enhancing execution efficiency.

While the TRL framework effectively improves adaptability and convergence, there is room for improvement, particularly in adapting to high-mobility devices and optimizing resource utilization with varying task granularity. Thus, while the TRL framework presents an effective solution for dynamic task offloading, further enhancements are necessary for its practical implementation. [19]

3) C-D3QN Algorithm: The C-D3QN algorithm combines K-means clustering with deep reinforcement learning (DRL) to optimize task offloading in dynamic environments. It uses a centralized decision-making agent to collect global state information, ensuring globally optimal decisions. This centralized framework enables efficient coordination among devices, addressing the challenge of suboptimal local decisions common in traditional methods.C-D3QN leverages K-means clustering to partition devices, reducing the action space and enhancing exploration efficiency, especially with a larger number of devices or tasks. By compressing the decision space, it facilitates more effective resource allocation, particularly when task requirements or device resources are complex.The reward function in C-D3QN accounts for energy consumption and

penalties, dynamically adjusting offloading decisions based on task load and device location, thereby optimizing resource allocation. This flexibility allows the system to adapt to environmental changes, improving overall efficiency.

Despite its benefits, C-D3QN is limited by its reliance on a coarse-grained task model, hindering fine-grained offloading optimization. It also lacks support for continuous action space models, which limits its applicability in some scenarios. Future research should incorporate fine-grained task models and policy gradient algorithms to further optimize resource utilization and performance. [20]

4) ML-RL: The ML-RL system enhances task offloading efficiency by leveraging collaborative agents across edge, fog, and cloud layers. Each layer is responsible for tasks with varying latency and computational power requirements: the edge layer handles low-latency tasks, the fog layer processes medium-latency tasks, and the cloud layer manages tasks requiring high computational power. This collaboration allows lower-layer agents to delegate offloading decisions to upper-layer agents, ensuring globally optimal decisions when local knowledge is insufficient.

The ML-RL system integrates greedy methods with reinforcement learning to improve offloading efficiency and convergence speed. Greedy methods, while providing suboptimal solutions through heuristic rules, rely heavily on real-time information, which can result in unstable execution. In contrast, reinforcement learning enables edge devices to improve their offloading strategy over time by interacting with the environment, maximizing long-term rewards. However, the system's inability to support task decomposition and parallel processing limits its applicability in complex scenarios. Enhancing task granularity processing and exploring more flexible policy learning algorithms would further improve system performance. [21]

5) MRLCO: MRLCO is an innovative task offloading solution leveraging Meta-Reinforcement Learning (MRL) and sequence-to-sequence (seq2seq) neural networks. The core innovation of MRLCO lies in its ability to quickly adapt to new environments with minimal data, reducing training time and computational overhead. This makes it significantly more efficient than traditional reinforcement learning methods.

The method models task offloading as multiple Markov Decision Processes (MDPs) and transforms the decision-making into a sequence prediction problem. Seq2seq networks are used to enhance decision flexibility, while an attention mechanism addresses information loss, improving decision-making accuracy and robustness. This flexibility allows MRLCO to adapt to dynamic challenges such as network fluctuations and device disconnections, ensuring accurate offloading decisions.Additionally, MRLCO features an adaptive client selection algorithm that filters underperforming clients, improving system stability and resource utilization. This ensures high offloading performance even in unstable environments.

Despite its advantages, MRLCO's performance still falls short of the optimal solution. Future research should explore more efficient non-strategic MRL techniques to further enhance decision-making and training efficiency. [22]

To further optimize existing methods, it is essential to develop fine-grained task offloading models. For instance, task decomposition based on Directed Acyclic Graphs (DAGs) could enhance decision-making capabilities when combined with policy gradient learning. [20] Additionally, research should focus on addressing the impact of high device mobility and dynamic network environments on task offloading performance to improve algorithm adaptability in real-world scenarios [18], [19]. Lastly, multi-objective optimization techniques should be prioritized to balance delay, energy consumption, and resource utilization, ultimately enhancing offloading efficiency and system stability in edge computing environments. [20]

B. Compression of Action Space and State Space

With limited resources of mobile devices, the action space and state space compression are crucial for efficiently managing the complexity of the task offloading problem in cloud-edge computing environments. Compressing these spaces simplifies the decision-making process,making it computationally feasible. And a smaller state space helps machine learning models generalize better, improving reliability. Besides, compression is key to managing increased complexity without overwhelming resources as system grows, and reducing the action space helps in creating a more unified approach to handle diverse devices and tasks. Here are some detailed methods of how these spaces are compressed:

1) Definition of Action Space and State Space: In the context of edge offloading, the action space typically encompasses the choices of executing tasks either on local devices or offloading them to various edge nodes, and The state space typically includes information such as the current state of the tasks, the state of the edge nodes (e.g., resource utilization), and network conditions (e.g., transmission delay, bandwidth)..This definition is mentioned in this paper, where the action space is defined as $A = \{taskMD_1, taskES_1, taskCS_1, \ldots, taskMD_n, taskES_n, taskCS_n\}$, indicating that tasks can be executed on mobile devices (MD), edge servers (ES), or cloud servers (CS). And the state space is defined as S , which includes computational capabilities F , data transmission rates V , and task scheduling $DEPt$ [18]. The papers addresses the challenge of a large action space by:

2) Clustering and Preprocessing: A Clustering-based Deep Reinforcement Learning Algorithm(C-D3QN)is proposed. By using clustering algorithms to preprocess the action space,terminal devices are divided into multiple device clusters,with each cluster taking the same action as a whole.This reduces the action space from$(M+1)^N$to$(M+1)^k$, where$N$is the number of terminal devices, $k$is the number of cluster centers,and$M$is the number of edge servers [20]. The clustering algorithm groups terminal devices with similar characteristics together,and the state of each device cluster can be represented by the state of the cluster center,thereby reducing the number of states that need to be processed and lowering the dimensionality of the state space.

Besides, by merging multiple workflows into one large workflow and assigning sub-deadlines to each task when modeling

a problem, the representation of the state space becomes more concise,thereby reducing the complexity of the state space [23].This compression significantly reduces the action space that needs to be explored, thereby improving the efficiency of the algorithm.

3) Constraints: Task Dependency Constraints: Considering the dependencies between tasks,where certain tasks can only be executed after others are completed,reduces the number of invalid action choices,thereby compressing the action space. It was mentioned that due to task dependencies,a task can only be executed after receiving the results of all its predecessor tasks,which implicitly compresses the action space [18].
Resource Limitation Constraints: Based on the resource limitations of edge nodes(such as computational power and memory capacity),excluding action options that do not meet the task's resource requirements further narrows down the action space. It was mentioned that the resources of edge clouds(such as CPU resources, memory, and bandwidth)are limited,and task offloading decisions need to consider these resource limitations,thus compressing the action space [19].
Time Window Restriction: Focusing only on the current and a future time window of state information,ignoring states that are too distant in the past,helps to reduce the size of the state space. [18]

4) Markov Modeling: Define states,actions,and rewards: States(S):Include key information such as device load,network latency,and resource utilization.Compress the state space to a manageable size through feature extraction and state clustering. Actions(A):Include operations such as task offloading,resource allocation,and service migration.Simplify the action space through action abstraction and hierarchical structuring. Rewards(R):Define the reward function based on indicators such as task execution time,energy consumption,and user satisfaction.For example,completing a task on time with low energy consumption can earn a high reward. Transition probabilities(P):Define the probability of transitioning to the next state after taking a specific action in a given state.These probabilities can be estimated using historical or simulated data. Policy(π):Define the actions to be taken in each state.Dynamic programming,reinforcement learning,and other methods can be used to optimize the policy,maximizing the expected sum of rewards. [22]

IV. Current Challenges and Future Prospects

A. Challenges

The application of reinforcement learning (RL) for edge computing task offloading encounters several challenges in practical settings. These challenges primarily involve the design of scenarios, algorithmic complexity, mobility issues, device and network heterogeneity, security concerns, and communication interference, among others.

1) Scenario Design Challenges: One of the primary challenges in applying RL to edge computing task offloading lies in scenario design [24]. While numerous studies rely on simulations and analyses based on static scenarios, real-world scenarios are inherently dynamic. Factors such as device mobility, network bandwidth fluctuations, and variations in user behavior can affect task loads, revealing the limitations of static offloading decisions. In contrast, RL methods, known for their adaptability in dynamic environments, offer a promising solution through the implementation of adaptive strategies. However, the ability to effectively model and adapt to environmental changes, ensuring that offloading decisions can be made in real time without introducing delays or inefficiencies, remains a critical problem that requires further development.

2) Algorithm Complexity and Latency Problem: The complexity of RL algorithms and the associated latency are significant challenges in edge computing task offloading [25]. In time-varying environments, RL often requires substantial computational resources and time to explore the high-dimensional action space, leading to slow processing speeds and delayed convergence during training. This issue is particularly problematic in real-time edge computing applications, where latency is a critical concern. Overly complex algorithms can increase computation times for offloading decisions, undermining system responsiveness. Therefore, the development of more efficient RL algorithms that reduce computational overhead while maintaining offloading effectiveness is crucial to improving the overall efficiency of task offloading in edge computing systems.

3) Mobility Challenges of Terminal Devices: The mobility of terminal devices presents significant challenges, particularly when these devices move across different service areas, requiring seamless task offloading and system stability [26], [27]. Devices may experience network fluctuations, bandwidth variations, and other issues during movement, which can negatively impact offloading efficiency. RL has the potential to address these challenges by enabling dynamic adjustments to offloading strategies. However, a critical issue remains the ability to maintain both timeliness and accuracy in offloading decisions, especially in environments characterized by device mobility and network fluctuations. Ensuring the adaptability and robustness of RL models is essential to enable effective task offloading in diverse network conditions.

4) Heterogeneous Network and Device Compatibility Issues: Heterogeneous device and network environments represent a significant obstacle for RL-based task offloading in edge computing [27]. The rapid proliferation of new devices and network topologies, driven by advancements in IoT and 5G technologies, has rendered traditional offloading strategies less effective due to the diversity of devices and networks. RL-based offloading solutions must address these compatibility issues to ensure the effectiveness of the offloading process. Developing adaptable and compatible offloading schemes that maintain efficiency across a wide variety of devices and network environments is a key challenge in current RL research for edge computing.

5) Security Challenges: Security remains a critical concern in edge computing task offloading [25]–[27]. The distributed nature of edge computing environments introduces vulnerabilities that can be exploited by malicious actors, especially in multi-tenant settings where system failures at a single point can lead to cascading effects. Ensuring the security of data during the offloading process is paramount to prevent data leakage or tampering. The integration of security measures into

RL-based offloading decision-making processes is essential to mitigate potential risks and enhance system resilience against adversarial attacks. Addressing these concerns is crucial for the practical deployment of RL in edge computing systems.

6) Communication Interference and Resource Management: The increasing number of devices in edge computing environments has led to a rise in network interference, which exacerbates the challenges of task offloading due to resource contention and communication congestion [26]. RL offers a promising approach to mitigate these issues by enabling dynamic resource allocation. However, a major challenge remains in optimizing resource distribution across multiple devices and complex network environments to ensure successful task offloading and improve quality of service. Developing RL algorithms that effectively balance offloading, resource allocation, and interference management is essential to maintain system stability and efficiency.

The challenges outlined above are inherently linked to the practical implementation of RL-based task offloading in edge computing. Addressing these issues will significantly enhance the feasibility and efficiency of RL-based task offloading systems. Solving these challenges not only provides theoretical grounding but also offers technological advancements, paving the way for broader adoption of edge computing technologies in practical applications.

B. Future Prospects

1) More Efficient Algorithm Design: With the increasing complexity of edge computing environments, future research can explore more efficient algorithms to address edge offloading issues. For instance, by integrating advanced artificial intelligence technologies such as deep learning and reinforcement learning, intelligent algorithms can be developed that can quickly adapt to environmental changes and make accurate offloading decisions. A paper mentions that future work will use game theory methods to solve dynamic and complex offloading problems in cloud-edge scenarios involving multiple devices and mobile sensing,which implies the exploration of more efficient algorithms [18]. Another paper aims to apply the MRLCO framework to other decision-making problems in MEC systems, such as content caching. Content caching in MEC aims to cache popular content on MEC hosts to achieve high service quality and reduce network traffic. The MRL framework can adapt to this problem by performing ”outer loop” training on cloud servers to learn meta-caching strategies, and ”inner loop” training on MEC hosts to learn specific caching strategies for each MEC host. [23]

2) Multi-objective Optimization: In practical applications,edge offloading needs to consider not only the execution delay of tasks but also multiple objectives such as energy consumption, cost, and service quality.Future research can further explore multi-objective optimization methods to maximize the overall benefits of task offloading. The objective of the task offloading problem is to minimize the total utility of each task, which involves the optimization of multiple objectives including task delay and energy consumption [19].

3) Cross-layer Collaborative Optimization: Edge computing systems consist of multiple layers of resources and networks,such as the device layer, edge layer, and cloud layer. Future research can consider cross-layer collaborative optimization to break down barriers between layers,achieving the global optimal allocation of resources and efficient task offloading. The task offloading problem in the cloud-edge environment mentioned in a paper implies the need for cross-layer collaborative optimization [18].

4) Dynamic Environmental Adaptability: The tasks and resource states in edge computing environments are dynamically changing.Future research can enhance the dynamic adaptability of algorithms,enabling them to quickly adjust offloading strategies in real-time changing environments to ensure the continuity and stability of tasks. A paper emphasizes the need for algorithms to adapt to dynamic cloud-edge environments,indicating a focus on dynamic real-world environmental adaptability [18]. Another paper mentions that they will implement the solution in an online setting, considering network dynamics and wireless interference and develop a real prototype to demonstrate the effectiveness of the proposed solution. [28]

5) Security and Privacy Protection: In the process of edge offloading,the security and privacy of data are important issues.Future research can focus on how to strengthen data security transmission,storage,and processing while ensuring the efficiency of task offloading,preventing data leakage and malicious attacks.Although security and privacy protection are not directly mentioned in the documents,they are important research directions in the field of edge computing.

V. Conclusion

This survey deeply explores the application of reinforcement learning in edge computing task offloading. Edge computing task offloading encounters numerous difficulties, and reinforcement learning has emerged as a powerful tool to address these issues.

DQN and its improved version D3QN lay the foundation for task offloading decisions. Methods such as Q-learning, D3QN, and MARL achieve optimized task offloading in different scenarios by reasonably defining the state and action spaces and designing reward mechanisms.

In terms of performance improvement, methods like DOQO and TRL have made improvements in aspects such as real-time performance and training cost, but they also have limitations. The compression of action and state spaces reduces computational complexity.

In practical applications, reinforcement learning faces challenges such as high algorithm complexity and latency, device mobility, heterogeneous compatibility, security, and resource management. Looking ahead to future research directions:

First, focus on developing more efficient algorithm systems, deeply integrating deep learning, reinforcement learning, and other advanced artificial intelligence technologies to create intelligent algorithms that can quickly respond to dynamic environmental changes and accurately make offloading decisions.

Second, deeply explore multi-objective optimization strategies, comprehensively coordinating multiple factors such as task execution delay, energy consumption, cost, and service quality to maximize the overall benefits of task offloading.

Third, fully promote the cross-layer collaborative optimization process, breaking the resource and network barriers among different layers of the edge computing system and constructing a collaborative ecosystem with globally optimal resource allocation.

Fourth, emphasize strengthening the dynamic environmental adaptability of algorithms to ensure their stable and efficient operation in complex and dynamically changing environments.

Through in-depth research and breakthroughs in these key areas, reinforcement learning is expected to achieve significant technological leaps in the field of edge computing task offloading, effectively promoting the wide application and in-depth integration of edge computing technology in numerous fields, providing solid technical support for various emerging application scenarios, accelerating the construction and development process of the intelligent interconnected world, and leading the innovation and transformation trend in the digital age.

REFERENCES

[1] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

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

[3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[4] X. Chen and G. Liu, "Energy-efficient task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge networks," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 843–10 856, 2021.

[5] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, p. 1054, 1998.

[6] S. Nath and J. Wu, "Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems," *Intelligent and Converged Networks*, vol. 1, no. 2, pp. 181–198, 2020.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] H. V. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2010, pp. 1–8.

[9] V. Mnih and D. Silver, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016, pp. 1928–1937.

[10] A. Robles-Enciso and A. F. Skarmeta, "A multi-layer guided reinforcement learning-based tasks offloading in edge computing," *Computer Networks*, vol. 220, p. 109476, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622005102

[11] P. Fraga-Lamas, S. Lopes, and T. Fernández-Caramés, "Green iot and edge ai as key technological enablers for a sustainable digital transition towards a smart circular economy: An industry 5.0 use case," *Sensors*, vol. 21, no. 17, p. 5745, 2021. [Online]. Available: http://dx.doi.org/10.3390/s21175745

[12] D. R. Morales and H. Romeijn, "The generalized assignment problem and extensions," in *Handbook of Combinatorial Optimization*, D.-Z. Du and P. Pardalos, Eds. Germany: Springer, 2005, vol. B, pp. 259–311.

[13] D. Balouek-Thomert, E. Renart, A. Zamani, A. Simonet, and M. Parashar, "Towards a computing continuum: Enabling edge-to-cloud integration for data-driven workflows," *International Journal of High Performance Computing Applications*, vol. 33, pp. 1159–1174, 2019.

[14] Z. Zhang, H. Li, Z. Tang, D. Gu, and J. Zhang, "A clustering offloading decision method for edge computing tasks based on deep reinforcement learning," *New Generation Computing*, vol. 41, pp. 85–108, 2023. [Online]. Available: https://doi.org/10.1007/s00354-022-00199-7

[15] X. Huang, S. Leng, S. Maharjan, and Y. Zhang, "Multi-agent deep reinforcement learning for computation offloading and interference coordination in small cell networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9282–9293, 2021.

[16] R. Zhang, F. R. Yu, J. Liu, T. Huang, and Y. Liu, "Deep reinforcement learning (drl)-based device-to-device (d2d) caching with blockchain and mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6469–6485, 2020.

[17] W. Hou, H. Wen, H. Song, W. Lei, and W. Zhang, "Multiagent deep reinforcement learning for task offloading and resource allocation in cybertwin-based networks," *IEEE Internet of Things Journal*, vol. 8, no. 22, pp. 16 256–16 268, 2021.

[18] X. Chen, S. Hu, C. Yu, Z. Chen, and G. Min, "Real-time offloading for dependent and parallel tasks in cloud-edge environments using deep reinforcement learning," *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, vol. 35, no. 3, pp. 391–404, MAR 2024.

[19] K. Shuai, Y. Miao, K. Hwang, and Z. Li, "Transfer reinforcement learning for adaptive task offloading over distributed edge clouds," *IEEE TRANSACTIONS ON CLOUD COMPUTING*, vol. 11, no. 2, pp. 2175–2187, APR-JUN 2023.

[20] Z. Zhang, H. Li, Z. Tang, D. Gu, and J. Zhang, "A clustering offloading decision method for edge computing tasks based on deep reinforcement learning," *NEW GENERATION COMPUTING*, vol. 41, no. 1, pp. 85–108, MAR 2023.

[21] A. Robles-Enciso and A. F. Skarmeta, "A multi-layer guided reinforcement learning-based tasks offloading in edge computing," *COMPUTER NETWORKS*, vol. 220, JAN 2023.

[22] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, vol. 32, no. 1, pp. 242–253, JAN 1 2021.

[23] L. Pan, X. Liu, Z. Jia, J. Xu, and X. Li, "A multi-objective clustering evolutionary algorithm for multi-workflow computation offloading in mobile edge computing," *IEEE Transactions on Cloud Computing*, vol. 11, no. 2, pp. 1334–1351, 2023.

[24] Y. Tian, Z. Liu, K. Zhang, Z. Li, and Y. Xie, "A survey on task offloading techniques in cloud-edge resource collaboration," *Computer Science and Exploration*, vol. 17, no. 10, pp. 2325–2342, 2023.

[25] J. Lv, J. Zhang, Z. Zhang, and C. Gan, "A survey on offloading strategies in mobile edge computing," *Mini-Micro Computer Systems*, vol. 41, no. 09, pp. 1866–1877, 2020.

[26] R. Xie, X. Lian, Q. Jia, T. Huang, and Y. Liu, "A survey on mobile edge computing offloading techniques," *Journal of Communications*, vol. 39, no. 11, pp. 138–155, 2018.

[27] Y. Zhang, Y. Liang, M. Yin, H. Quan, T. Wang, and W. Jia, "A survey of computation offloading schemes in mobile edge computing," *Journal of Computer Research and Development*, vol. 44, no. 12, pp. 2406–2430, 2021.

[28] Y. Sahni, J. Cao, L. Yang, and Y. Ji, "Multi-hop multi-task partial computation offloading in collaborative edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1133–1145, 2021.