Knowledge and Information Systems (2021) 63:2479–2511
https://doi.org/10.1007/s10115-021-01590-4

REGULAR PAPER

Deep reinforcement learning-based resource allocation and seamless handover in multi-access edge computing based on SDN

Chunlin Li[1,2] · Yong Zhang[1] · Youlong Luo[1]

Received: 8 September 2020 / Revised: 20 June 2021 / Accepted: 27 June 2021 /
Published online: 21 July 2021

Abstract
With the access devices that are densely deployed in multi-access edge computing environments, users frequently switch access devices when moving, which causes the imbalance of network load and the decline of service quality. To solve the problems above, a seamless handover scheme for wireless access points based on perception is proposed. First, a seamless handover model based on load perception is proposed to solve the unbalanced network load, in which a seamless handover algorithm for wireless access points is used to calculate the access point with the highest weight, and a software-defined network controller controls the switching process. A joint allocation method of communication and computing resources based on deep reinforcement learning is proposed to minimize the terminal energy consumption and the system delay. A resource allocation model is based on minimizing terminal energy consumption, and system delay is built. The optimal value of task offloading decision and resource allocation vector are calculated with deep reinforcement learning. Experimental results show that the proposed method can reduce the network load and the task execution cost.

Keywords Multi-access edge computing · Seamless handover · Software-defined network · Deep reinforcement learning

1 Introduction

With the development of wireless technology and the Internet of Things, applications need to consume a large number of network and computing resources to reduce low latency [1]. However, the limited computing power, battery life, and storage space of mobile devices

✉ Chunlin Li
lichunlin74@126.com

[1] Department of Computer Science, Wuhan University of Technology, Wuhan 430063, People's Republic of China

[2] Industrial Software Engineering Technology Research and Development Center of Jiangsu Education Department, Nanjing Institute of Industry Technology, Nanjing, People's Republic of China

can hardly meet the above requirements. To solve this problem, multi-access edge computing (MEC) is proposed to expand the applicability of Wi-Fi and multiple fixed access technologies in heterogeneous networks [2]. MEC is deployed at the edge of the mobile network by service providers and has the characteristics of low latency, low energy consumption, and perception. The user's request is directly processed by the local MEC, which greatly reduces the end-to-end delay and the backhaul network load. However, MEC still faces many challenges to meet the needs of users for high data rates and high computing capabilities, such as user mobility issues, resource coordination management issues, security and billing issues, and privacy protection issues. Software-defined Network (SDN) is a new type of network architecture that separates the control plane and forwarding plane of network equipment. It breaks the vertical coupling of the network architecture and simplifies the network configuration and policy implementation [3]. MEC hides the complexity of the heterogeneous edge computing environment from end-users by the programmability of SDN, which ensures the effective implementation of services [4–6].

In the SDN-based MEC environment, the intensive deployment of access equipment leads to frequent handovers and ping-pong effects of users during the movement, which results in a waste of wireless channel resources and a decline in the quality of experience (QoE). Therefore, it will become very important to realize the seamless handover of mobile terminals at multiple network access points and to guarantee the quality of service during handover. To provide various services, newly developed applications require more and more resources, which poses challenges to current mobile devices, which are often limited in terms of computing, storage, and battery capacity due to portability. How to properly configure communication resources and computing resources in SDN-based MEC environment will be of great significance to achieve computing load balancing, reduce task computing delay and terminal energy consumption. Focusing on the realization of SDN-based MEC environment, the seamless switching of terminal devices between network access points, the joint management of communication and computing resources, we carry out two aspects of research: based on quantum particles group-optimized seamless switching method for wireless access points, and DRL-based joint management method of communication and computing resources. The main contributions are summarized as follows.

1. In order to solve the problem of frequent handover and ping-pong effect caused by the dense deployment of small cells in the SDN-based MEC environment, a seamless switch scheme for wireless access points based on load awareness is proposed. This scheme monitors the network status in real time by the centralized control function of the SDN controller, which considers the traffic load of AP and the load of MEC, and the real-time change of signal reception strength. SDN migrates the routing logic to the central controller, which uses real-time global information to make forwarding decisions and configure forwarding rules on the switch. In SDN, intelligent management logic is installed on the controller, which greatly simplifies the operation and management of the network. The instantaneous changes caused by geographically dispersed services and the dynamic behavior of the network can also be captured. This algorithm can ensure task continuity during handover and reduce network load.
2. In order to reduce the task's calculation delay and terminal energy consumption, a joint allocation method of communication and computing resources based on DRL is proposed. First, a resource allocation model based on minimizing terminal energy consumption and system delay is built. Then, the optimal value of task offloading decision and resource allocation vector are calculated with deep reinforcement learning. The joint

resource allocation method can reasonably allocate the communication and computing resources in MEC based on SDN, which is of great significance to ensure the service quality of mobile terminal services. At the same time, it can reduce the network load and the transmission delay of communication link, improve the service distribution ability and improve the resource utilization. This strategy can obtain the best results of task offloading and resource allocation, and reduce task execution costs.

3. Experimental results show that the seamless handover algorithm can ensure task continuity during handover and reduce network load. The resource allocation method obtains the optimal value of task offloading and resource allocation, and reduces the cost of task execution.

The rest of this paper is structured as follows. Section 2 reviews related works and Sect. 3 presents the SDN-based seamless switching and resource management scheme in MEC environment. Section 4 presents the implementation of the algorithms. Section 5 evaluates the implementation and validates the advantages of our approaches. Finally, conclusions and future work are given in Sect. 6.

2 Related work

2.1 SDN-based MEC architecture

Mahdi et al. [5] proposed a secure and energy-saving SDN controller blockchain support architecture, an efficient distributed trust authentication scheme. Zhang et al. [7] analyzes the combination of blockchain and SDN to ensure the trust of network platform and effective network management. Huang et al. [8] proposed and implemented a low-latency MEC framework based on SDN, which provided flexible data plane programmability for MEC and mobile core networks while maintaining compatibility with Third-Generation Partnership Project (3GPP). Katov et al. [9] proposed a method for resource consolidation toward minimizing the power consumption in an extensive network, with a substantial resource over-provisioning. Schiller et al. [10] developed an SDN controller to manage the traffic on the MEC system. The controller is used for the evaluation of public security solutions. Peng et al. [11] applied SDN control functions to achieve intelligent flow control and effective multi-resource management, which reduced the cost of function configuration. Miladinovic et al. [12] introduced SDN and MEC technologies to transmit and process data generated by many sensors in smart cities, which achieved optimal distribution of intelligent city applications between MEC servers and remote cloud computing centers. Xia et al. [13] proposed a new industrial IoT architecture with a layered SDN controller and MEC-based radio access network (RAN). Wang et al. [14] proposed a new DRL-based offloading framework, which can efficiently learn the offloading policy uniquely represented by a neural network. Hamid et al. [15] proposed a new and trustable framework for MEC management systems with crucial security and authentication components by which it ensures the delivery of users' QoE. Yazdinejad et al. [16] proposed a secure and energy-saving blockchain support architecture for the SDN controller of the IoT, making the blockchain suitable for IoT devices with limited resources.

In this paper, we study seamless handover and resource allocation of MEC architecture based on SDN, ensuring the continuity of users' services before and after handover to reduce the network load. The optimal decision vector and resource allocation vector can be obtained, which reduces the cost of task execution.

2.2 Seamless handover

Bao et al. [17] proposed a framework that supports a job pre-migration mechanism, which can migrate computing jobs when switching is expected. Yunoki et al. [18] proposed the ongoing service is restored with a new appropriate MEC server, which can achieve centralized control of the network and support users' high mobility. Neto et al. [19] proposed a taxonomy that provides a comprehensive view of the reasons for solution deployment and the scenarios in which the solution runs. Bi et al. [20] proposed a fog computing architecture, which improves the handoff performance and data communication efficiency in mobile fog computing. Zeljković et al. [21] proposed a modular handover management framework based on SDN, which reduce the delay in the handover process and the signaling overhead of handover. Yin et al. [22] proposed a new fast switching scheme for SDN-based vehicle network to improve the switching performance.

Mouawad et al. [23] proposed an SDN-based vehicle network mobility management solution, which provided handover expectations to ensure low latency and seamless handover. Zhang et al. [24] proposed a seamless handover authentication protocol rush for 5 g heterogeneous networks to ensure the security and efficiency of frequent handover in 5 g wireless roaming environment. Mohseni et al. [25] proposed a handoff scheme based on hierarchical SDN, which reduces the number of handoffs in the mobile process. Bi et al. [26] proposed a SDN-based mobility management (SDN-MM) solution to decouple mobility management and packet forwarding.

Through the analysis of a large number of research results, it can be seen that although the existing switching schemes ensure the continuity of service, their considerations are often not careful enough, ignoring the impact of bandwidth, memory, and CPU load of the edge server on the switching, which leads to a decrease in network performance because of the unnecessary frequent handovers.

2.3 Resource management

Zhong et al. [27] proposed a new task popularity estimation scheme, which effectively reduces the delay and energy consumption. Lyu et al. [28] proposed a heuristic unloading decision algorithm, which can maximize the utility of the system. Tran et al. [29] studied the problem of joint task offloading and resource management in order to maximize the user's task offloading benefits. Cheng et al. [30] considered a multi-user and multi-MEC server system based on orthogonal frequency division multiplexing access, in which task offloading strategies and wireless resource management were jointly studied. Liang et al. [31] studied energy-efficient resource management in software-defined mobile networks with mobile edge computing and caching capabilities. Wang et al. [32] studied distributed wireless transmission resources between the MEC server and the underlying edge device, and distributed wired transmission resources between the cloud center and the MEC server to provide low-latency services for IoT real-time applications. Qian et al. [33] proposed two algorithms to determine the best grouping of the edge to serve offloaded computing workloads of different intelligent terminals. Yang et al. [34] invoked the Markov decision process for resource management problems of mobile users and access points to learn the best offloading and resource management strategies from historical experience.

Through the analysis of a large number of research results, it can be seen that although the existing resource management scheme improves the system performance to a certain extent [27–37], they do not consider the impact of SDN on network resource management. For the terminal device, due to the uncertainty of user movement and wireless conditions, the joint management of communication resources and computing resources is crucial for user task processing.

3 Seamless handover scheme of wireless access point based on load awareness

3.1 Seamless handover model of wireless access point based on load awareness

The system framework of the SDN-based MEC environment is shown in Fig. 1, which is mainly divided into an infrastructure layer, an edge computing layer, and a cloud computing layer. The key components include mobile devices, base stations, wireless access points (AP), SDN controllers, MEC servers, and cloud data centers. The infrastructure layer comprises various hardware devices such as mobile devices, base stations, wireless APs, switches, and optical fibers. Various access technologies make up different access networks. For example, the wireless access network composed of base stations and wireless APs has higher flexibility and mobility. The fiber access network composed of optical fibers, switches, and gateways provides users with high-bandwidth services. The MEC server is deployed near the cellular base station and the wireless AP in the edge computing layer. Multiple MEC servers can collaborate and complete tasks through backhaul links to provide better service for mobile users by balancing workload and sharing resources. The

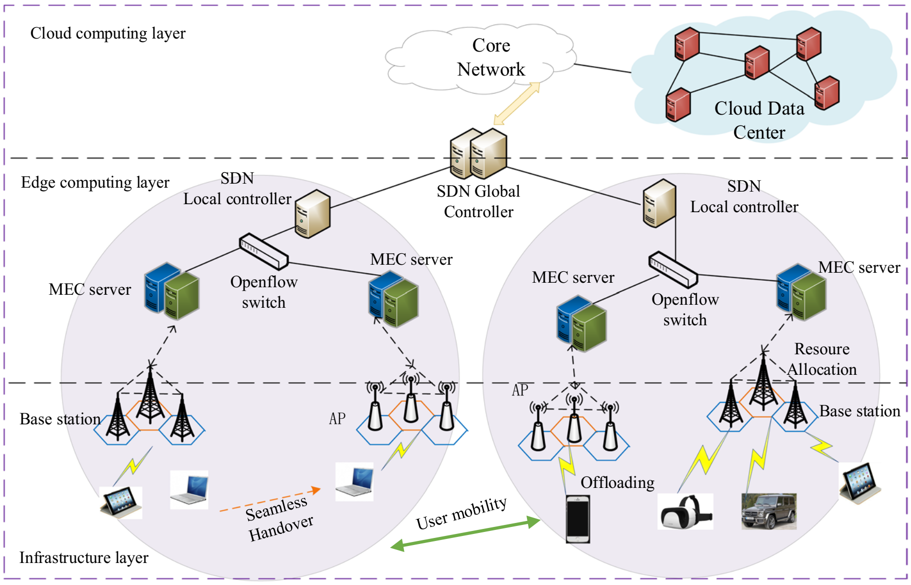


Fig. 1 System framework of SDN-based MEC environment

MEC server and the SDN controller communicate through the OpenFlow protocol. The SDN local controller controls each control domain, and the controller can poll the switch to collect statistics about the active flow. Alternatively, it can request traffic statistics to be pushed at a specific frequency (when the traffic exceeds the time limit). The controller has a global view of the network. These functions of the OpenFlow controller can be used to develop complex and effective monitoring solutions. In this way, SDN local controller can form a global view of network topology and resources. The local controller of SDN collects information such as nodes, paths, and bandwidth to abstract network resources. The global controller of SDN is considered to be the engine of the entire MEC. All the main controls and decisions are initialized and related to processing in the global controller. The centralized SDN architecture can distribute the overall control load among the local controllers and reduce the overhead of the global controller. When the resources at the edge computing layer are limited, or when processing tasks insensitive to delay and computationally intensive, tasks can be offloaded to the cloud data center for execution.

To balance the load between APs in the MEC environment and alleviate the ping-pong effect of frequent mobile terminal handover between APs, this paper uses the centralized control function of the SDN controller to form a global view of the network topology during network initialization. Each controller adds an application that adds mobility allocation capabilities to the controller. The SDN controller monitors the network status of each AP in real-time and collects user terminal information so that the AP only has the data forwarding function. To select access points in the seamless handover, this paper comprehensively considers the traffic load of AP and the load of MEC, and the real-time change of signal reception strength. It proposes a seamless handover algorithm based on load awareness. Among them, the weighted sum of CPU utilization, memory utilization, and bandwidth utilization is taken as the load of the MEC server. The quantum particle swarm optimization algorithm is used to solve the model, and the access point with the most significant access weight is decided. According to the handover factor compared with the current access point, the SDN controller decides whether to perform the handover. The seamless handover process considered in this paper is different from the long-distance service migration scheme and only considers adjacent access points. The data packets sent during the handover process include signal strength messages and load conditions. The data size is minimal, so the cost of task migration is low and can be ignored [38]. The seamless handover scheme of wireless access points based on load awareness is shown in Fig. 2.

3.1.1 SDN-based seamless handover communication model in MEC environment

An SDN-based MEC environment includes multiple access technologies. For example, macro-base stations and micro-base stations are used as network access points to provide services for users. When users move in an SDN-based MEC environment, multiple users have different types of execution of the terminal service, which are connected to the AP through a wireless link and communicate with the MEC server. In an SDN control domain of the access plane, let $M = \{m_1, m_2, ..., m_N\}$ represents the set of MEC servers in the system, $A = \{a_1, a_2, ..., a_N\}$ represents the set of APs in the system, $U = \{u_1, u_2, ..., u_K\}$ represents a collection of users in the system. Any one of these users can only connect to a single AP at a time. $F_{i,j}$ is the user connection matrix, which indicates the connection between the user and the AP. $F_{i,j} = 1$ indicates that the user u_i is connected to the AP a_j, and $F_{i,j} = 0$ indicates that the user u_i is not connected to the AP a_j. Let U_{aj} denote the set of users managed by AP a_j.

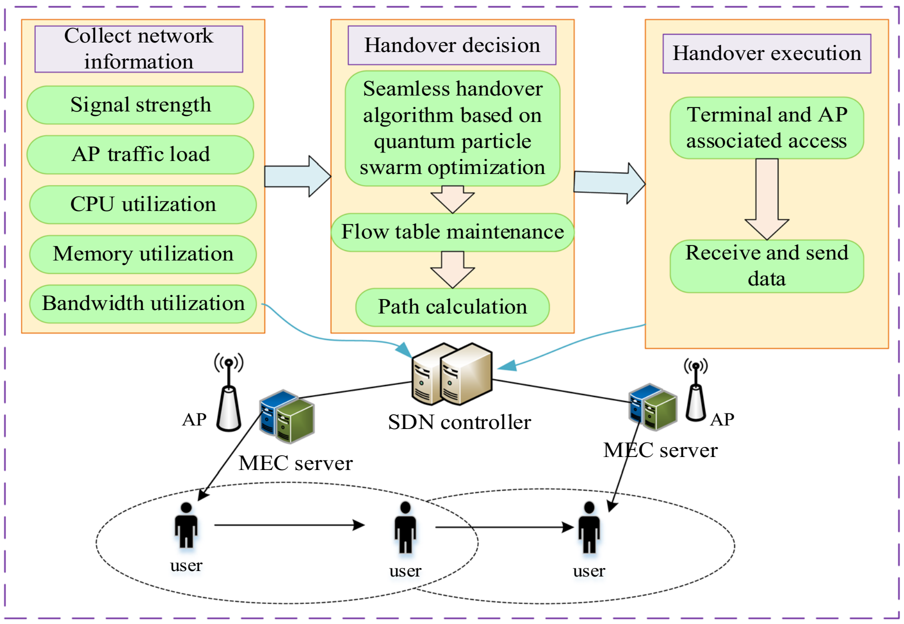


Fig. 2 Seamless handover scheme of wireless access point based on load awareness

The clients can continuously generate up-link traffic, and each user is allowed to connect to only one AP at the same time. The downlink traffic for AP transmissions can be ignored. In this paper, logarithmic distance path loss model is used as channel transmission model. Signal strength is an indispensable factor in seamless switching, and it has a positive correlation with the channel anti-jamming ability in data transmission. The received signal strength between the access point a_j and the user u_i is defined as follows [39].

$$P_{dij} = P_{tx} + G_t + G_r - 20\log_{10}(4\pi f d_{\mathrm{ref}}\sqrt{L}/c) - 10\delta\log_{10}(d_{ij}/d_{\mathrm{ref}}) \tag{1}$$

where P_{tx} represents the transmit power of user u_i, G_t is the transmit gain of user u_i, G_r is the receive gain of access point a_j, d_{ij} is the distance from u_i to a_j, d_{ref} is the reference distance for calculating loss, f indicates frequency band, L indicates system loss, c indicates speed of light, and δ indicates path loss factor.

Therefore, the uplink signal-to-interference-plus-noise ratio (SINR) can be expressed as follows [40].

$$\mathrm{SINR}_{i,j} = \frac{P_{dij}}{I_r + \sigma^2} \tag{2}$$

where $I_r = \sum_{q\in K,k\notin i} P_{dij} (1 \le q \le K,\ 1 \le j \le N)$ is the cumulative interference generated by all users that are sending data except u_i, and σ^2 is additive white Gaussian noise.

3.1.2 AP and edge server load model

3.1.2.1 AP load The traditional handoff between different APS in the same network is divided into two stages: trigger handoff and handoff execution. The handoff trigger stage refers to that with the continuous movement of sta in the network, the communication quality between sta and the current AP is getting worse and worse, resulting in the current wireless connection becoming more and more unstable, thus triggering the handoff process. There are three main processes in the handoff execution stage. Firstly, the network scan is carried out to obtain the status information of each AP in the network, and the handoff target AP is selected. Then the workstation sends the connection request to the selected handoff target AP for identity authentication. Finally, the workstation sends the association request to the handoff target AP to reestablish a new wireless connection. The allocation usually establishes the connection with AP according to the signal strength, and lacks the consideration of the real load of AP, which makes the problem of low utilization of network resources and unbalanced load of access point in traditional network structure more and more serious.

In this paper, the traffic load of the AP is used as the AP load model. Let $L_{\max} = \{L_{1,}L_{2,}L_3 \cdots L_N\}$ represent the maximum load of the AP set. When the traffic load of the AP exceeds the maximum value, it means that the AP is overloaded and the connection relationship between the AP and the user needs to be redeployed. Because of the communication between user u_i and AP a_j. Within a period of time I, user u_i receives I_{AI} bytes and sends O_{AI} bytes, and the interval between two acquisitions is T, then the average traffic of user u_i is expressed as follows.

$$\overline{V_{A,n}} = \begin{cases} \sum_{I=1}^{n} (I_{AI} + O_{AI}) & n \geq 1 \\ nT & n = 0 \end{cases} \tag{3}$$

The instantaneous flow rate of user u_i is expressed as follows.

$$V_{A,n} = \begin{cases} \frac{(I_{An}+O_{An})}{T} & n \geq 1 \\ 0 & n = 0 \end{cases} \tag{4}$$

Each user terminal has different requirements for communication quality. For relatively balanced flow, the change of instantaneous flow is small, and the average flow can be used to predict the estimated flow rate of the user terminal; for sudden and relatively strong data flow, the flow estimate of the user terminal should be predicted by the instantaneous flow. The flow of user u_i can be expressed as follows.

$$V_A = V_{A,n} \cdot \varepsilon + \overline{V_{A,n}} \cdot (1 - \varepsilon) \tag{5}$$

where $\varepsilon < 1$, for relatively balanced service flows, the value of ε is small, and for data services with relatively strong bursts, the value of ε is large. The traffic of all user terminals connected by a_j is:

$$L_{a_j} = \sum_{i \in U_{aj}} V_i \tag{6}$$

The load condition of the access point is expressed as follows.

$$Load_{a_j} = \frac{L_{a_j}}{L_j} \tag{7}$$

3.1.2.2 MEC load This paper considers the service type to calculate the incoming traffic which is distributed to the MEC server. According to the CPU, memory, and bandwidth utilization parameters, the load of each MEC server Load_{m_j} can be calculated as.

$$\begin{aligned} &\mathrm{Load}_{m_j} = \alpha \times \mathrm{CPU}_r + \mu \times \mathrm{MEM}_r + \beta \times \mathrm{Band}_r \\ &s.t.\ \alpha + \mu + \beta \ = \ 1\ \alpha, \mu, \beta \in (0, 1) \end{aligned} \tag{8}$$

where α, μ, β represents the weight factors of CPU, memory and bandwidth utilization in the MEC server, respectively. Different types of user requests have different weight values for CPU, memory, and bandwidth.

3.1.3 Modeling of seamless handover based on load awareness

How the user terminal selects the appropriate AP as the new access point is related to whether the load is balanced among APs. We have taken SINR, AP load, and MEC server load into consideration and introduce a new access evaluation weight $\mathrm{W}_{i,j}$, which is defined as follows.

$$W_{i,j} = \frac{\mathrm{SINR}_{i,j}}{1 + \mathrm{Load}_{a_j} + \mathrm{Load}_{m_j}} \tag{9}$$

Therefore, the optimization goal is:

$$\begin{aligned} &\max W_{i,j} \\ &s.t\, C1\ :\ \alpha + \mu + \beta = 1 \quad \alpha, \beta, \mu \in (0, 1) \\ &C2\ :\ L_{a_j} < L_j, \quad \forall j \\ &C3\ :\ \textstyle\sum_i F_{i,j} = 1, \quad \forall j \in \{1, 2, ..., N\} \end{aligned} \tag{10}$$

where $\mathrm{W}_{i,j}$ represents the weight of the user u_i associated with the AP a_j, the larger the value, the more resources available on the AP, and the user terminal should access the AP. Constraint $C1$ means that the total weight of the user request for each resource in the MEC server is 1, and constraint $C2$ means that the load of each AP cannot exceed its threshold, so as to avoid overloading AP traffic and causing various access points in the network unbalanced load. Constraint $C3$ means that a user can only connect to one AP at a time.

If the access weights of the candidate AP and the original AP are similar, the terminal device will frequently switch. This situation not only aggravates the workload of the SDN controller, but also degrades the user's service quality, and is not good for the network load balancing. In order to avoid this situation, this paper defines a switching factor. When the switching factor is greater than the system switching factor threshold, the switching is performed. The switching factor threshold is set to different values according to the different needs of users. The calculation formula of the switching factor $H_{a,b}$ is defined as follows.

$$H_{a,b} = \frac{W_a - W_b}{W_b} \tag{11}$$

where W_a and W_b are the weights of the candidate AP and the original AP, respectively. In this way, the switch decision made by the SDN controller can reduce the workload of the controller and improve the network performance.

3.2 Seamless handover algorithm of wireless access point based on load awareness

In this paper, a quantum-behaved particle swarm optimization algorithm is introduced to solve the switching problem of wireless access points based on load awareness. Searching for the global optimal solution, which is the access point with the largest $\mathrm{W}_{i,j}$. In the QPSO algorithm, each particle represents a candidate solution. Each particle is randomly initialized in the d-dimensional search space. The dynamic behavior of particles is very different from the dynamic behavior of particles in the classical PSO system. A single particle cannot know the precise values of position and velocity at the same time. The state of a particle is described by the wave function $\Psi(\vec{x}, t)$ rather than the position and velocity in the PSO algorithm. In the wave function $\Psi(\vec{x}, t)$, $\vec{x}$ is the position of the particle and t is time.

The total number of particles is Q and the current position vector of particle k ($k = 1, 2..., Q$) is represented as $\vec{x}_k$, which is defined as follows:

$$\vec{x}_k = (\vec{x}_k^1, \vec{x}_k^2, ..., \vec{x}_k^i, ..., \vec{x}_k^K) \tag{12}$$

where $\vec{x}_k^i$ represents the connection between the i-th user and the AP, which is defined as follows:

$$\vec{x}_k^i = (F_{i,1}, F_{i,2}, ...F_{i,N}) \tag{13}$$

Each particle is updated according to the following rules:

$$\begin{cases} \vec{x}_k(n+1) = P + \lambda|E(n) - \vec{x}_k(n)| \cdot \ln(1/\eta), & h \geq 0.5 \\ \vec{x}_k(n+1) = P - \lambda|E(n) - \vec{x}_k(n)| \cdot \ln(1/\eta), & h < 0.5 \end{cases} \tag{14}$$

where n represents the total number of iterations,λ is the expansion and contraction factor. η, h is a random number from 0 to 1, and $E(n)$ is the average optimal position, which is defined as follows.

$$E(n) = \frac{\sum_{k \in Q} P_k(n)}{Q} \tag{15}$$

The position update function P is defined as follows.

$$P = \chi P_k(n) + (1 - \chi)G(n) \tag{16}$$

where $P_k(n)$ represents the optimal position of the k-th particle at the n-th iteration, $G(n)$ represents the global optimal position, which is selected from $P_k(n)$ of all particles according to the fitness function, which is defined as follows.

$$J(F_{i,j}) = f(F_{i,j}) - \chi P_f(F_{i,j}) \tag{17}$$

where $f(F_{i,j})$ is the objective function, χ is the penalty factor, and $P_f(F_{i,j})$ is the penalty function, which is defined as follows.

$$P_f(F_{i,j}) = \left[\max(0, L_{aj} - L_j)\right]^2 + \sum\nolimits_{i\in K} (\sum_{j\in N} F_{i,j})^2 + \sum_{j\in N}\sum_{i\in K} (F_{i,j}^2 - F_{i,j})^2 \tag{18}$$

The optimal position of the k-th particle at the n-th iteration is defined as follows:

$$P_k(n) = \begin{cases} \vec{x}_k(n), & \text{if } J[\vec{x}_k(n)] > J[P_k(n-1)] \\ P_k(n-1), & \text{if } J[\vec{x}_k(n)] \le J[P_k(n-1)] \end{cases} \tag{19}$$

$G(n)$ is the global optimal solution, which is defined as follows.

$$\xi = \arg \max_{1\le k\le Q} \{J[P_k(n)]\} \tag{20}$$

$$G(n) = P_\xi(n) \tag{21}$$

According to the above solution process, the core pseudocode of the wireless access point seamless switching algorithm based on quantum-behaved particle swarm optimization proposed in this paper is shown in Algorithm 1.

Algorithm 1: Seamless handover algorithm for wireless based on quantum-behaved particle swarm optimization

```
Input: Initialize the maximum number of iterations $N$, the particle swarm size $Q$, the initial position of each particle $\vec{x}_k(0)$
Output: Global optimal position $G(n)$
1: $P_k(0) \leftarrow \vec{x}_k(0)$
2: $G(0) \leftarrow P_k(0)(k = 1,2,...,Q)$ // According to formula (32), select the global optimal position from the local optimal position
3: $n \leftarrow 0$
4: while $n \le N-1$ do
5:     $E(n) \leftarrow \sum_{k\in Q} P_k(n)/Q$ , $P \leftarrow \chi P_k(n) + (1-\chi)G(n)$ // Calculate the average optimal position according to formula (30), and calculate the position update function according to formula (31)
6:     Update $\vec{x}_k(n)$ // Update the position of each particle according to formula (29)
7:         if $J[\vec{x}_k(n+1)] > J[P_k(n)]$ then // Update $P_k(n)$ according to fitness function
8:             $G(n+1) \leftarrow P_k(n+1)$
9:         else $P_k(n+1) \leftarrow P_k(n)$
10:        if $J[P_k(n+1)] > J[G(n)]$ then// Update $G(n)$ according to fitness function
11:            $G(n+1) \leftarrow P_k(n+1)$
12:        else $G(n+1) \leftarrow G(n)$
13         n++
14 end while
15: return $G(n)$
```

The specific implementation steps are as follows:

1. Initialize the maximum number of iterations N, the particle swarm size Q, the initial position of each particle $\vec{x}_k(0)$.

2. Evaluate the fitness of all particles, and retain the individual optimal position and global optimal solution.
3. Update the particle population according to the particle update rules.
4. If the maximum number of iterations is reached, the optimal position of the group is output, otherwise it returns to step 2.
5. The SDN controller decides whether to switch based on the switching factor calculation formula.

In the design of the quantum-behaved particle swarm optimization algorithm, the particle population size is Q, the maximum number of iterations of the particle is N, each particle is randomly initialized in the d-dimensional search space, the particle state is described by the wave function $\Psi(\vec{x}, t)$, And the particle has quantum behavior that can improve the global convergence ability of the algorithm, and its time complexity is $O(Q \times N \times d)$.

4 DRL-based joint allocation scheme of communication and computing resources

4.1 DRL-based joint allocation model of communication and computing resources

Communication and computing resources are vital for the offload and execution of computing tasks by edge users. The former determines the data rate and energy consumption during transmission. The latter limits the computation time and execution energy of tasks offloaded to the MEC server. How to allocate communication resources and computing resources in MEC based on SDN properly is of great significance for ensuring the service quality of mobile terminal services. It can reduce network load and transmission delay of communication links, improve service distribution capabilities, and increase resource utilization. When exploring resource allocation of MEC based on SDN, the core task is to determine whether the task needs to be offloaded and whether offloading can reduce task response delay and terminal energy consumption. In other words, our goal is to allocate communication resources and computing resources to mobile devices in a reasonable manner to minimize the task execution cost of all users in the system. This section combines factors such as terminal energy consumption and system delay for task execution to solve the above problems, shown in Fig. 3.

4.1.1 Communication resource model

Assuming that the number of user devices in this section is N and the number of MEC servers is M, user tasks can be executed locally or offloaded to the MEC server. This chapter sets that if multiple tasks use the same channel, when data are transmitted through wireless resources, the tasks will interfere with each other, resulting in a reduction in the data transmission rate. The set of user equipment in a single cell is represented as $N = 1, 2, \dots, n$. Assuming that each user has delay-sensitive or computationally intensive tasks that need to be processed, the tasks can be executed locally or offloaded to the MEC server via a wireless link for execution. The set of MEC servers is expressed as $M = \{1, 2, ...m\}$, and the capacity of MEC servers is limited, which may not be enough for all users to offload tasks to the edge server for execution. Define W as the bandwidth of the wireless channel. If multiple users choose to perform tasks at the same time, the wireless bandwidth will

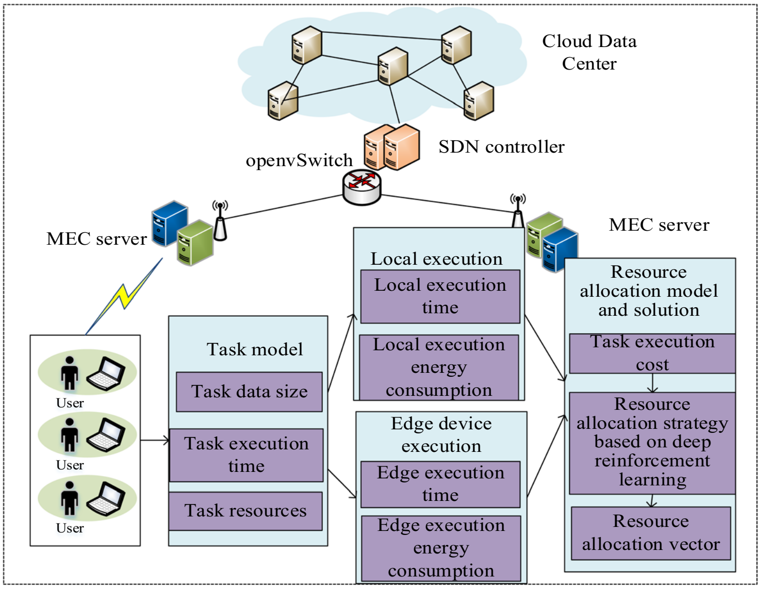


Fig. 3 DRL-based joint allocation scheme of communication resources and computing resources

be allocated to the user to upload data. Assuming that user k and MEC server m establish a connection through a wireless link, define the bandwidth ratio as $\beta_k^m(0 \le \beta_k^m \le 1)$, the upload rate r_k is given as follows.

$$r_k = W\beta_k^m \log_2\left(1 + \frac{P_k|G_k^m|^2}{\sigma_k^2}\right) \tag{22}$$

where P_k is the transmission power used by user k to upload data, $|G_k^m|^2$ is the channel gain of the wireless access point near user k and the MEC server m in the wireless channel, and σ_k^2 is the power of Gaussian noise.

4.1.2 User task model

Assuming that the task that user k needs to perform is $R_k = \{B_k, D_k, \gamma_k\}$, which can be offloaded to the MEC service m or to execute locally. Let B_k represents the input data of task R_k, which includes the input parameters and application code of the task. D_k is the computing resource required to complete the task R_k, which is specifically defined as the number of CPU cycles required to complete the task. It can be seen that the size of D_k and B_k are positively related. γ_k represents the maximum allowable delay of task R_k, which means that each user's experience delay should not exceed γ_k.

Assuming that tasks cannot be subdivided, which means that each user should perform their tasks through local calculations or offload calculations. For resources available on the MEC server $m \in M$ (such as computing resources, communication resources, and storage resources), the user $k \in N$ can obtain its MEC server resources through the wireless channel. Let the offload decision variable $\alpha_k = \{0, 1\}$, $\alpha_k = 1$ means that the task R_k of user k is executed locally, and $\alpha_k = 0$ means that the task R_k is offloaded and executed on the MEC server m. Let offload decision vector $A = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ represent the execution decision of the user task.

4.1.3 Computing resource model

4.1.3.1 Local computing model If user k chooses to execute task R_k locally, then let T_k^l as the task local execution time of user k, which only includes the processing delay of the terminal device CPU. The number of CPU cycles per second, f_k^l is defined as the local computing power of user k. The local computing power will vary between different user devices. The local execution delay of task R_k is defined as follows.

$$T_k^l = \frac{D_k}{f_k^l} \tag{23}$$

Let E_k^l represent the local energy consumption for performing task R_k:

$$E_k^l = D_k z_k \tag{24}$$

where z_k represents the energy consumption of each CPU cycle that completes task R_k, $z_k = 10^{-11}(f_k^l)^2$. Combining formula (23) and formula (24), it can be concluded that the total cost of calculated locally C_k^l is given as follows.

$$\begin{aligned} & C_k^l = w_k^t T_k^l + w_k^e E_k^l \\ & s.t\ 0 \le w_k^t \le 1 \\ & 0 \le w_k^e \le 1 \\ & w_k^t + w_k^e = 1 \end{aligned} \tag{25}$$

where w_k^t and w_k^e represent the delay weight and energy consumption weight of executing task R_k respectively. The weight setting can standardize the delay and energy consumption of different units. Under different criteria, it can reflect the user's tendency to offload tasks, such as more sensitive to delay or more sensitive to energy consumption. When the user is in a low power state and pays more attention to energy consumption, w_k^t is closer to 0 and w_k^e is closer to 1. When the user has sufficient power and is running some delay-sensitive applications (such as video distribution, augmented reality), w_k^t is closer to 1, and w_k^e is closer to 0.

4.1.3.2 MEC server calculation model If user k chooses to perform its task R_k on the MEC server m, the entire calculation process will be divided into three steps:

1. User k needs to upload enough input data (program code and parameters) to the AP through the wireless access network, The AP forwards the data to the MEC server m.
2. The MEC server m will allocate resources instead of user k to perform task R_k.
3. The MEC server m returns the execution result to user k.

Let $T_{k,t}^m$ represent the transmission delay for user k to transmit input data to MEC server m, which is defined as follows.

$$T_{k,t}^{m} = \frac{B_k}{r_k} \tag{26}$$

where r_k is the upload rate of user k, and B_k represents the size of calculation input data required by task R_k.

Let $T_{k,p}^{m}$ represents the processing delay of MEC server m executing task R_k, which is defined as follows.

$$T_{k,p}^{m} = \frac{D_k}{f_k} \tag{27}$$

where f_k is the computing resource allocated by the MEC server m to the computing task R_k, such as the number of CPU cycles per second, and D_k represents the total number of CPU cycles required to complete the computing task R_k.

Let $E_{k,t}^{m}$ represent the transmission energy consumption of user k transmitting input data to MEC server m, which is defined as follows.

$$E_{k,t}^{m} = P_k T_{k,t}^{m} = \frac{P_k B_k}{r_k} \tag{28}$$

Let F define the total computing resources of MEC server m, the total amount of resources meets the condition as follows.

$$\sum_{k=1}^{n} (1 - \alpha_k) f_k \le F \quad \forall k \in N \tag{29}$$

Assuming that when MEC server m is performing task R_k, user k is idle. This paper define the power of user k in the idle state as P_k^i, the energy consumption of user k in the process of MEC server m performing tasks $E_{k,p}^{m}$ can be calculated as follows.

$$E_{k,p}^{m} = P_k^i T_{k,p}^{m} = \frac{P_k^i D_k}{f_k} \tag{30}$$

After the task R_k is executed on the MEC server m, the transmission delay $T_{k,d}^{m}$ of the user's download result data is defined as follows.

$$T_{k,d}^{m} = \frac{B_d}{r_d} \tag{31}$$

where B_d is the size of the resulting data, r_d is the download speed. However, the download data rate is usually high, and the size of the resulting data is much smaller than the size of the input data, so this article omits the delay and energy consumption of the process.

According to formulas (27) and (31), the delay T_k^m for user k to offload task R_k to MEC server m is given as follows.

$$T_k^m = T_{k,t}^{m} + T_{k,p}^{m} \tag{32}$$

According to formulas (28) and (30), the energy consumption E_k^m of user k offloading task R_k to MEC server m can be obtained as follows.

$$E_k^m = E_{k,t}^{m} + E_{k,p}^{m} \tag{33}$$

Combining formula (32) and formula (33), which can be concluded that the cost C_k^m of offloading task R_k to MEC server m by user k is given as follows.

$$C_k^m = w_k^t T_k^m + w_k^e E_k^m \tag{34}$$

Then the total cost C_{all} of all users performing tasks collectively is given as follows.

$$C_{\text{all}} = \sum_{k=1}^{N} \alpha_k C_k^l + (1 - \alpha_k) C_k^m \tag{35}$$

The resource allocation problem in SDN-based MEC considers the task offload decision of multiple users and the joint allocation of communication and computing resources. The optimization goal of the model is to minimize the execution cost of user tasks, the delay of task execution and the terminal energy consumption, the construction issues are defined as follows.

$$\begin{aligned}
&\min \sum_{k=1}^{n} \alpha_k C_k^l + (1 - \alpha_k) C_k^m \\
&s.t.\ C1 : \alpha_k \in \{0, 1\}, \quad \forall k \in N \\
&C2 : (1 - \alpha_k) T_k^m + \alpha_k T_k^l \le \gamma_k, \quad \forall k \in N \\
&C3 : \sum_{k=1}^{n} (1 \text{ - } \alpha_k) f_k \le F, \quad \forall k \in N \\
&C4 : 0 \le f_k \le (1 \text{ - } \alpha_k) F, \quad \forall k \in N \\
&C5 : 0 \le \beta_k^m \le 1, \quad \forall k \in N
\end{aligned} \tag{36}$$

Among them, constraint $C1$ means that the task can only be offloaded to the MEC server for execution or local execution; constraint $C2$ means that the execution delay of the task cannot exceed its maximum allowable delay; constraint $C3$ means that it is assigned to all offload tasks to the MEC server The total computing resources of the executed user set do not exceed the total computing resources of the MEC server; constraint $C4$ means that the computing resources allocated to a single user do not exceed the total computing resources of the MEC server. Constraint $C5$ means that the bandwidth allocated to a single user does not exceed the total bandwidth that the communication link can provide.

4.2 DRL-based joint allocation algorithm of communication and computing resources

4.2.1 State space and action set

In the resource allocation problem of SDN-based MEC environment, the state space s consists of two parts. Define $s = \{C_{\cos t}, C_r\}$, where $C_{\cos t}$ is the weighted value of the system environment user task execution delay and energy consumption, that is $C_{\cos t} = C_{\text{all}}$,where C_r is the available resource capacity of MEC server m, which is calculated as follows.

$$C_r = F - \sum_{k=1}^{n} (1 - \alpha_k) f_k \quad \forall k \in N \tag{37}$$

The agent selects an action from the action space based on the value of the current state s. If the task is offloaded to the MEC server for execution, the power and bandwidth are allocated to the user equipment. If users choose to execute locally, users will consume local power to perform the task. The action space $a = \alpha_1, \alpha_2, ..., \alpha_n$ consists of offloading decision vector A.

4.2.2 Reward function

In DRL, after the agent performs the corresponding action a according to the state s, the environment will grant a specific reward $r(s, a)$. Each state-action pair will have a $Q(s, a)$ value, which is choose the expected discount reward of action a from state s, this value can be regarded as a long-term reward. Usually, the reward function is related to the objective function. The objective function of this paper is to minimize the cost of user task execution, and the goal of DRL is to obtain the largest long-term reward. After each action, if we reduce the cost, we will get a positive reward; if we increase the cost, we will get a negative reward. Therefore, the value of the reward should be inversely related to the size of the objective function. In the system environment of this paper, instant reward $R_{\text{immediate}}$ is defined as follows.

$$R_{\text{immediate}} = 1 - \frac{C_{\cos t}(s, a)}{C_{\cos t}(\text{local})} \tag{38}$$

where $C_{\cos t}(\text{local})$ is the cost calculated locally by the task and $C_{\cos t}(s, a)$ is the cost of the current state. For user k, if the local calculation cannot satisfy the delay constraint, that is $T_k^l > \gamma_k$, then offload its task to execute on the MEC server.

Q-learning is a classic reinforcement learning scheme and a learning scheme to record Q value. Each state-action pair will have a $Q(s, a)$ value. For each step, the agent calculates $Q(s, a)$ and stores it in the Q table. This value can be regarded as a long-term reward, the $Q(s, a)$ can be expressed as:

$$Q(s_t, a_t) = Q(s_t, a_t) + \eta[r_{t+1} + \lambda \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \tag{39}$$

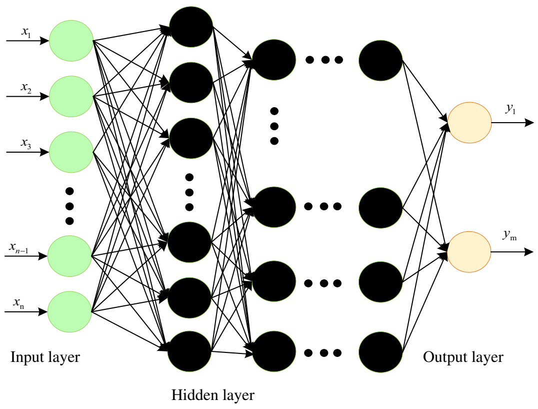
where $0 < \eta < 1$ represents the learning rate, $0 < \lambda < 1$ represents the attenuation factor, When λ approaching 0, it means that the agent focus on instant rewards, λ approaching 1, indicating that the agent is also very concerned about future rewards. The next step is a, and the next step is s_{t+1}.

4.2.3 Deep *Q* network

Artificial neural network generally has three layers of structure: input layer, hidden layer and output layer. The output end of each layer is set with corresponding activation function, and there are corresponding weights between layers. Data from the input layer into the neural network, after all the neurons in the network processing, the output of the neural network can be obtained. However, in order to improve the network's ability to depict data features, the number of hidden layers is usually increased. The structure of deep neural network (DNN) is shown in Fig. 4.

Although Q learning can solve the problem by obtaining the maximum return, if we use Q learning, it means that for each state-action pair, we need to calculate the corresponding Q value and store it in the Q table. But as the number of users increases and the scale of the

Fig. 4 Deep neural network



problem increases, the Q table will be very large. Therefore, we use the DNN to estimate $Q(s, a)$ instead of calculating the Q value for each state action pair, which is the basic idea of deep Q network (DQN).

In an SDN-based MEC environment, the problems of task offloading and resource allocation consist of two alternating stages, which includes generation of task offload decision and update of task offload decision. The generation of offloading decisions depends on the use of DNN, which is characterized by its embedded parameters θ, such as the weight of the hidden neurons connected. In the tth time slice, the DNN takes the current system state s as input, selects an optimal action a according to the objective function of the solution, the reward obtained by the system is $Q(s, a)$ and compares the newly obtained state-action pair (s, a) is added to the replay memory. Then, in the tth time slice offload scheme update phase, a batch of training samples is extracted from memory to train the DNN, and the parameter θ_t of the DNN is updated to θ_{t+1}. In the next time slice, the offload decision is updated according to the new system state. Thereafter, the operations of these two stages are repeated, and the task offload decision of DNN is gradually improved.

The neural network of this algorithm uses the ReLU function as the activation function of the hidden layer of the network. ReLU function is a nonlinear activation function, which is defined as follows:

$$\text{relu}(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{40}$$

After the Q value is output by the neural network, this article uses ε-greedy strategy to choose actions. Neural network training requires a loss function optimization process, which minimizes the deviation between the label and the output, and the parameters of the neural network will be updated by back propagation and gradient descent. The neural network in DQN is no exception. The goal of DQN is to make the Q value close to the target Q value and use the Q learning algorithm to provide so-called labels. In this article, we use mean square error (MSE) [41] as the loss function of the neural network in DQN, which can be defined as:

$$L(\theta) = E[(r + \lambda \max_{a_{t+1}} Q'(s_{t+1}, a_{t+1}, \theta') - Q(s_t, a_t, \theta))^2] \tag{41}$$

where $r + \lambda \max_{a_{t+1}} Q'(s_{t+1}, a_{t+1}, \theta')$ is the target Q value calculated by the parameters θ' of the target network, and $Q(s_t, a_t, \theta)$ is the evaluation Q value with the parameter θ output by the evaluation network. The target network is used to provide fixed labels and improve the stability and convergence of training. The initial parameters of the target network are the same as the evaluation network, but θ is updated every step, θ' is updated every C steps.

Algorithm 2 shows the core pseudocode of DRL-based joint allocation algorithm of computation and communication resources.

Algorithm 2: DRL-based joint allocation algorithm of computation and communication resources

```
Input: Number of iterations $M$, Attenuation factor $\lambda$, Action set $a$, Exploration factor $\varepsilon$.
Output: Optimal task offload decision vector $A = \{\alpha_1, \alpha_2, ..., \alpha_n\}$, And the corresponding resource allocation vector $f = \{f_1, f_2, ..., f_n\}$
1:  Initialize the replay memory buffer $D$
2:  Initialize the evaluation network, the random weight is $\theta$
3:  Initialize the target network, the weight is $\theta' = \theta$
4:  for episode=1, $T$ do
5:      Get current status $s$
6:      for $t = 1, 2, ..., M$ do
7:          Take the state $s$ as input to get the Q value output corresponding to all actions of the neural network
8:          Generate a random number $\upsilon \in [0,1]$
9:          if $\upsilon < \varepsilon$ then
10:             Randomly generate action $a_t$
11:         else select $a_t = \arg\max_{a_t} Q(s_t, a_t, \theta)$
12:         end if
13: Resource allocation according to action $a_t$
14: Get rewards $r_t$ and the next status $s_{t+1}$
15: Store experience $(s_t, a_t, r_t, s_{t+1})$ into replay memory buffer $D$
16: Randomly take samples from the replay memory buffer $D$ and calculate the current target Q value
17:         if episode ends at step t+1 then
18:             $y_t = r_t$
19:         else $y_t = r_t + \lambda \max_{a_{t+1}} Q'(s_{t+1}, a_{t+1}, \theta')$
20:         end if
21: Update gradient d of the evaluation network parameter $\theta$ according to formula (41)
22: Update the target network weight $\theta' = \theta$ every $C$ steps
23:     end for
24: end for
25: return $A = \{\alpha_1, \alpha_2, ..., \alpha_n\}$, $f = \{f_1, f_2, ..., f_n\}$
```

The specific ε implementation steps are described as follows:

1. Initialize replay memory buffer D, evaluate network, target network, weight, etc.
2. Get the current state s.
3. Take the state s as input to get the Q value output corresponding to all actions of the neural network.
4. Use ε-greedy strategy to choose action a_t.

5. Perform resource allocation according to action a_t to get reward r_t and next state s_{t+1}.
6. Store experience (s_t, a_t, r_t, s_{t+1}) into the replay memory buffer D, and randomly take samples from the replay memory buffer D to calculate the current target Q value.
7. Train the neural network according to the loss function.
8. Update the target network weight every C steps
9. Repeat steps 2–8 until the state space search is completed.

5 Evaluation

5.1 Experimental environment

5.1.1 Experimental setup

As shown in Fig. 5, the experimental environment consists of user equipment, an SDN controller, and an edge server. User equipment is the source of basic data information for the entire system, which can access the MEC server through a wireless access point. Two servers in different geographic locations represent edge nodes, and each edge node is a Docker container cluster environment, which uses Docker containers to provide flexible computing and storage resources. The edge node and the control node interact and forward data through the OpenFlow protocol. The control node integrates the SDN controller and

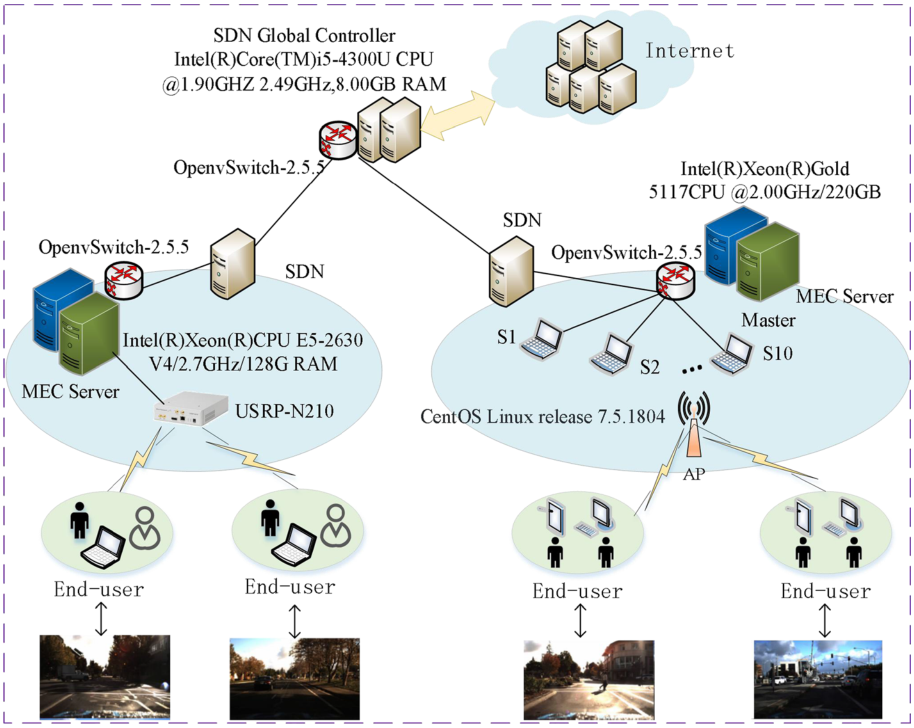


Fig. 5 Architecture of the experimental environment

the Kubernetes container orchestration tool, which provides communication to the network access device and edge nodes through the OpenFlow protocol of the southbound interface in the SDN architecture, which delivers flow table operation information for programmable network control. This experimental platform provides user terminal access through network access equipment, edge nodes provide high-bandwidth and low-latency application services, and the SDN controller implements network management.

5.1.2 Experimental benchmark dataset

The benchmark test dataset determines the correctness and reliability of the performance evaluation of the algorithm. The autonomous driving data set released by Udacity on Github [42] is selected. The data set includes 24,423 California street scene pictures, including trucks, cars, traffic lights, and pedestrians. The resolutions of the pictures are 1900×1200. Some sample pictures are shown in Fig. 7.

5.1.3 Performance metrics

To evaluate the feasibility of the load-aware wireless access point seamless handover scheme in SDN-based MEC environment, we select the average number of handovers, throughput, packet loss rate, load balance, average handover delay, and handover success rate as the algorithm performance evaluation indicators. The average number of handovers represents the number of handovers that occur during the movement of the terminal equipment of the entire system within a period time. The throughput is an important parameter reflecting the load situation. The packet loss rate can be calculated as $P\_\text{drop} = 1 - (1 - P_c) \times (1 - \text{PER})$. Load balance (LB) reflects the overall working status of the system at a certain moment which can be calculated as follows.

$$\text{LB} = \frac{\left(\sum B_i\right)^2}{\left(n \cdot \sum B_i^2\right)} \tag{42}$$

where B_i is the load of its edge server and n is the number of APs in the network. LB is a scalar in the interval [0, 1]. The average handover delay refers to the time it takes for data to travel from the sending end of the network to the receiving end during the handover. The handover success rate is the probability of a user's successful handover.

In order to evaluate the feasibility of DRL-based joint allocation scheme of communication and computing resources, this paper selects average service time, terminal energy consumption, task completion rate, and task execution cost as algorithm performance evaluation indicators. The average service time [43] is the average duration of all tasks in the system from generation to the final execution result, which is calculated as follows:

$$\text{AST} = \frac{\sum_{i=1}^{\text{SucR}} (\text{TRE}_i - \text{TRS}_i)}{\text{SucR}} \tag{43}$$

where the SucR is the number of tasks of successfully executed, the TRE_i is the completion time of first task, and the TRS_i is the start time of the first task. Terminal energy consumption refers to the power consumption of the mobile terminal from the generation to the final execution result of all tasks in the system. It also includes local execution energy consumption and offloading to the edge server transmission energy consumption. Task completion rate [44] refers to the ratio of the number of tasks whose response time meets the task

Table 1 Initialization configuration of main experimental parameters

Parameter	Value	Meanings of the parameter
K	4	Number of wireless channels owned by base station
σ^2	-100	Thermal noise power (DBM)
f_e	[1, 9]	CPU frequency of edge server (GHz)
f_n	[0.5, 2.0]	Local computing rate of mobile users (GHz)
P^T	[1, 3]	Power of user equipment in sending/receiving state (W)
P^b	[0.1, 0.5]	Power of CPU in idle state of user equipment (W)

Table 2 Initialization configuration of main experimental parameters

Parameter	Value	Meanings of the parameter
K	4	Number of wireless channels owned by base station
σ^2	-100	Thermal noise power (DBM)
f_e	[1, 9]	CPU frequency of edge server (GHz)
f_n	[0.5, 2.0]	Local computing rate of mobile users (GHz)
P^T	[1, 3]	Power of user equipment in sending/receiving state (W)
P^b	[0.1, 0.5]	Power of CPU in idle state of user equipment (W)
w^t/w^e	{0.1, 0.3, 0.5, 0.7, 0.9}	Task execution weight of delay/energy consumption
B_k	[50, 5000]	Average data size of upload/download (KB)

deadline to the total number of tasks, where task deadline is determined according to the urgency of the task and the execution capability of the system. Task execution cost is the weighted sum of user waiting time and energy consumption for task execution.

5.1.4 Experimental parameters

In check to see the correctness and effectiveness of the seamless handover algorithm for wireless access point based on load awareness, in SDN-based MEC environment, the spectrum bandwidth of the AP is set to 20 MHz [45] and that of BS access is set to 40 MHz [46]. In the experiment, mobile devices move back and forth between multiple wireless access points to switch. SDN controller collects network information of each terminal and AP. The main experimental parameters and initialization configuration are shown in Table 1.

In order to verify the correctness and effectiveness of DRL-based joint allocation algorithm of computation and communication resources, the spectrum bandwidth of the AP is set to 20 MHz and that of BS access is 40 MHz in SDN-based MEC environment. In the performance verification experiment of DRL-based joint allocation algorithm for communication and computing resources (DRL-RA), the situation that multi-users process computing tasks within 30 min is simulated. The task arrival mode of each device meets Poisson distribution. The main experimental parameters and initial configuration are given in Table 2.

5.1.5 Benchmark algorithms

To analyze the performance of the wireless access point seamless handover algorithm proposed in this paper based on load awareness, this paper selects three completely different seamless handover strategies as the comparison algorithms. Strongest signal first algorithm (SSF) [47] mainly considers that each terminal device in the network can receive the largest RSSI value, which can often obtain better performance indicators. The minimum load first algorithm (MLF) [48] is to select the access point with the least load to switch under the condition that the network transmission rate of the terminal is not lower than the threshold. This type of algorithm often only considers the load of the AP, not the load of the edge server. QoS-Based Handover algorithm (QBH) [49] mainly considers the utility of the user side. Although QBH improves the accuracy of decision-making, it does not consider the terminal service type, which will lead to the problem of unreasonable access point selection during the handover process.

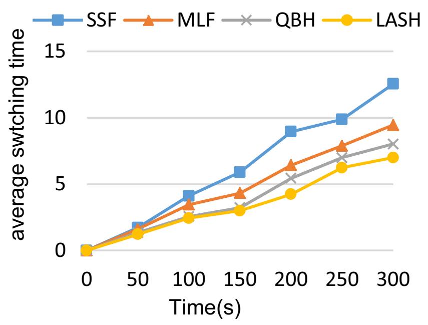
To analyze the performance of DRL-RA, this paper chooses three different resource allocation strategies as comparison algorithms. Minimize total processing completion time (MTPCT) algorithm [50] aims to minimize the total processing completion time of all tasks. However, it does not consider reducing the energy consumption of the user terminal as much as possible while shortening the task processing delay. Minimize terminal energy consumption (MTEC) algorithm [51] minimizes the energy consumption of all mobile terminals under the constraints of application waiting time. This algorithm can greatly improve the task completion efficiency of users with insufficient power. Cost and latency tradeoff algorithm (CLT) [52] token the cost of task uploading, task execution, and result transmission as offloading cost. Each user will solve his own optimization problem when controlling resource allocation to maximize his role.

5.2 Results and analysis

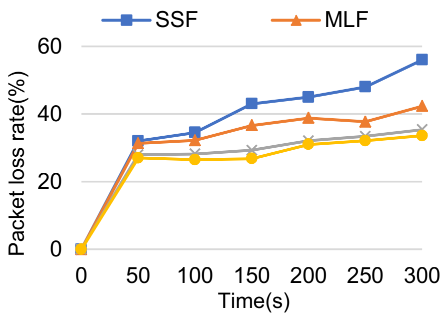
5.2.1 Experimental verification of seamless handover scheme of wireless access point based on load awareness

The seamless handover algorithm (LASH) proposed in this paper is compared with SSF, MLF, and QBH. The influence of the number of mobile users and the user's moving rate on the average number of handovers, throughput, packet loss rate, load balancing rate, average handover delay, and handover success rate over a period of time is considered.

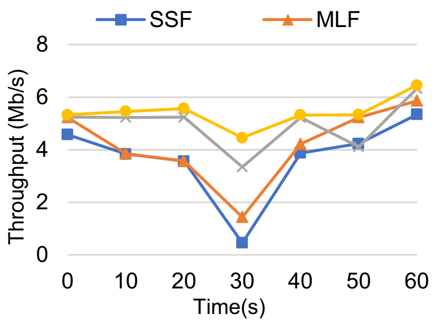
5.2.1.1 Change of experimental results with time Figure 6a–d shows the average switching times, packet loss rate, throughput, and switching delay over time. It can be seen from the figure that the average switching times and packet loss rate of the four algorithms increase with time. Due to frequent switching between SSF and MLF, service interruption is prone to occur, and the packet loss rate increases faster. QBH and LASH can consider a variety of factors to achieve seamless handover of user equipment, with fewer interruptions during the handover and lower packet loss rate. Besides, when handover occurs between SSF and MLF, the handover throughput decreases significantly. However, the switching between QBH and LASH is smoother, and the decrease in throughput is small. LASH omits the authentication phase in the handover process, which can greatly shorten the data transmission delay during the handover process.



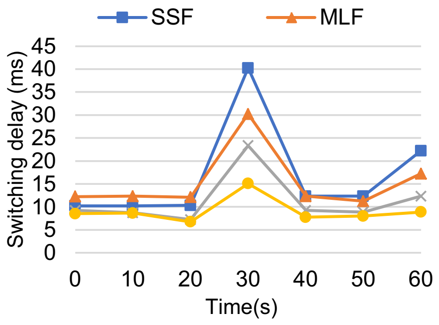
(a) Change of average switching time with times



(b) Change of packet loss rate with times



(c) Change of throughput with times

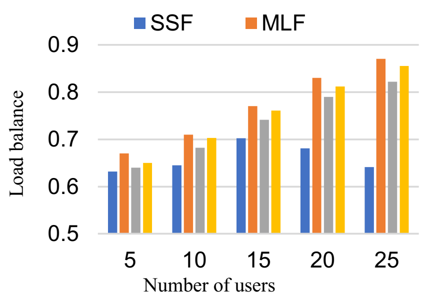


(d) Change of switching delay with times

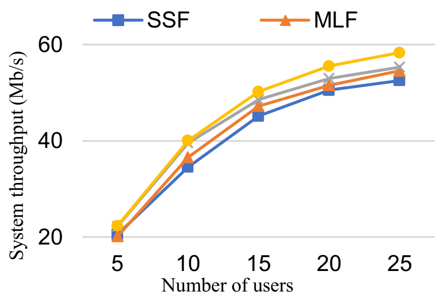
Fig. 6 The change of experimental results with time

5.2.1.2 The impact of the number of users on the experimental results Figure 7a–d, respectively, shows the change process of load balance, system throughput, packet loss rate, and handover success rate with the number of users. It can be seen from the figure that the number of handovers increases as the number of user increases, and the system throughput of the four algorithms gradually increases and the growth rate slows down. LASH considers the load situation between APs when switching and avoids load imbalance among APs. When the number of mobile users is 25, LASH is 33.39% higher than SSF in terms of load balance, 4.01% higher than QBH, and 1.72% lower than MLF. In terms of system throughput, LASH is 10.99% higher than SSF, 6.92% higher than MLF, and 5.42% higher than QBH. As the number of users increases, the packet loss rates of the four algorithms gradually increase, and the growth rate slows down, and the handover success rates of the four algorithms all decrease.

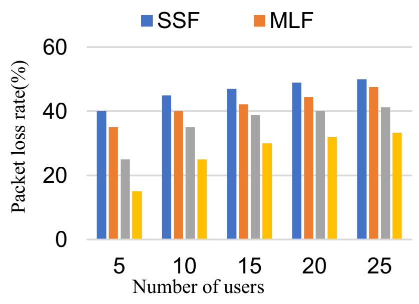
5.2.1.3 The effect of user movement speed on experimental results Figure 8a–d, respectively, shows the change process of handover delay, the average number of handovers, packet loss rate, and handover success rate with the user's moving rate. As the user's moving speed increases, the handover delay of the user equipment between APs also increases. The increase in the user's moving speed has led to an increase in the total number of user handovers in the system, and the packet loss rates of the four algorithms have gradually increased.



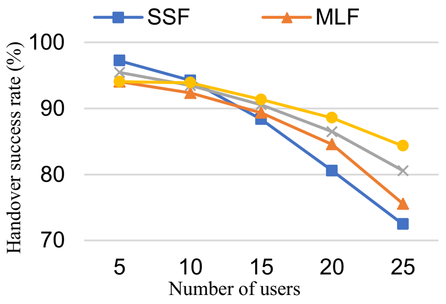
(a) Change of load balance with users



(b) Change of system throughput with users



(c) Change of packet loss rate with users



(d) Change of handover success rate with users

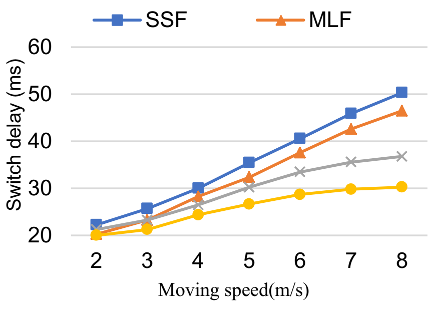
Fig. 7 The change of experimental results with the number of users

LASH considers handover based on multiple factors such as signal reception strength, AP load, and MEC load, which reduces the number of unnecessary handovers and makes the handover success rate decrease more smoothly as the user's moving speed increases. When the user's moving speed is 6 m/s, LASH is 11.9 ms lower than SSF, 8.9 ms lower than MLF, and 6.9 ms lower than QBH. In terms of average switching times, it is 46.10% less than SSF, 39.97% less than MLF, and 21.68% less than QBH. In terms of packet loss rate, it is 12.67% lower than SSF, 11.67% lower than MLF, and 6.34% lower than QBH. The handover success rate is 6.21% higher than SSF, 4.1% higher than MLF, and 0.66% higher than QBH.

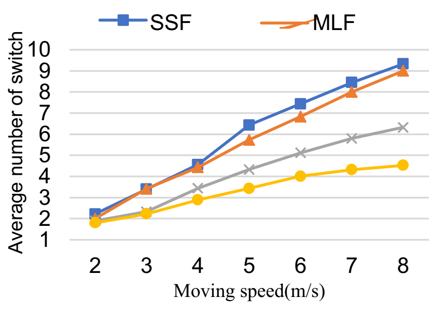
5.2.2 Experimental verification of DRL-based joint allocation scheme of communication and computing resources

The DRL-based joint allocation algorithm for communication and computing resources proposed in this paper is compared with MTPCT, MTEC, and CLT. The influence of the number of tasks, MEC server processing capacity, decision weight, and calculation task size on average service delay, terminal energy consumption, task completion rate, and task execution cost are also considered. This experiment uses the control variable scheme and still uses the target detection data set. Each experiment is repeated 20 times under the same conditions, and the average value is used as the final experimental result.

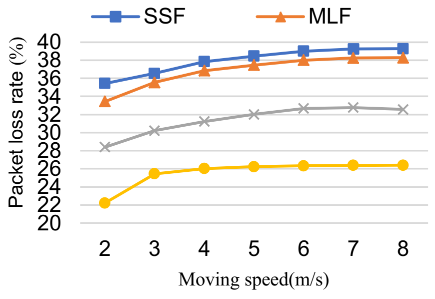
5.2.2.1 The impact of the number of tasks on the experimental results Figure 9a–d shows the average service delay, terminal energy consumption, task completion rate, task execu-



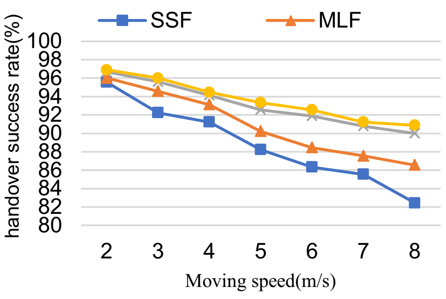
(a) Switch delay with moving speed



(b) Average number of switch with moving speed



(c) Packet loss rate with moving speed

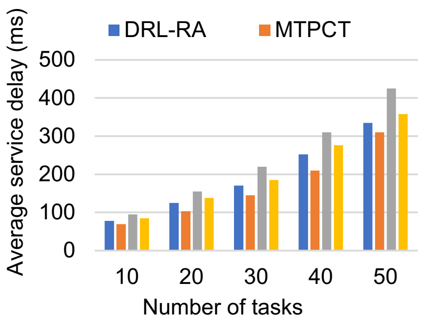


(d) handover success rate with moving speed

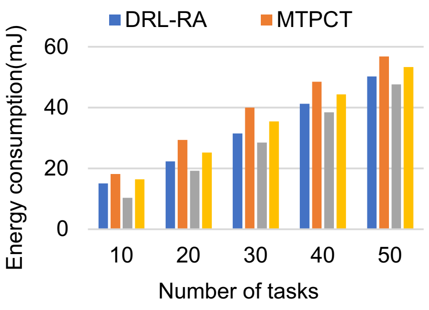
Fig. 8 The effect of user movement speed on experimental results

tion cost changes with the number of tasks. It can be seen from the figure that the average service delay, terminal energy consumption, and task execution cost increase as the number of tasks increases, and the task completion rate decreases as the number of tasks increases. DRL-RA runs a DRL-based resource allocation algorithm in an SDN-based MEC architecture and uses the centralized control function of SDN to allocate communication resources and computing resources reasonably, consider delay and energy consumption factors. It can be seen from the figure that when the number of tasks is 50, the average service delay of DRL-RA is reduced by 21.18% compared with MTEC, 6.42% lower than CTL, and 8.06% higher than MTPCT. In terms of terminal energy consumption, it is 11.64% lower than MTPCT, 5.41% higher than MTEC, and 5.85% lower than CLT.

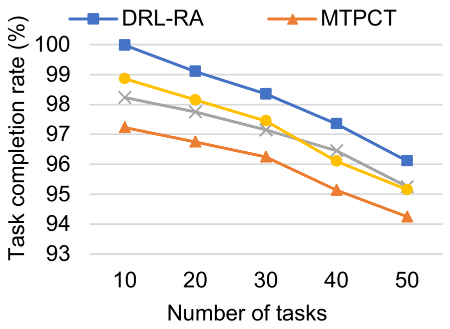
5.2.2.2 The influence of the processing capacity of the MEC server on the experimental results Figure 10a–d, respectively, shows the average service delay, terminal energy consumption, task completion rate, task execution cost, and MEC server processing capacity changes. The average service delay gradually decreases as the processing capacity of the MEC server increases. When the processing capacity of the MEC server is 8 GHz, DRL-RA is 1% lower than MTPCT, 8.76% lower than CLT, and 28% lower than MTEC. When the processing capacity of the MEC server reaches 5 GHz, the terminal energy consumption changes little and tends to be stable. When the MEC server processing capacity is 8 GHz, DRL-RA reduces the terminal energy consumption by 13.39% compared



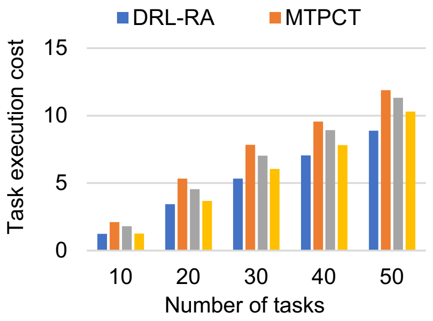
(a) Change of average service delay with tasks



(b) Change of energy consumption with tasks



(c) Change of task completion rate with tasks

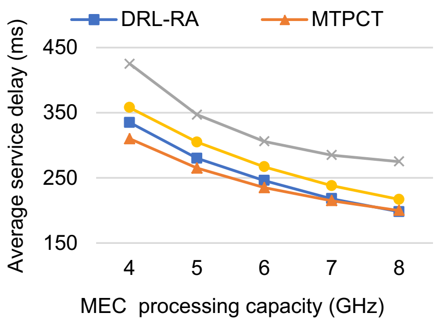


(d) Changes in task execution costs with tasks

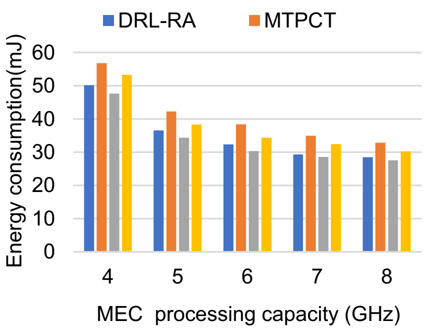
Fig. 9 The influence of the number of tasks on the experimental results

with MTPCT, reduces by 5.95% compared with CLT, and increases by 3.27% compared with MTEC. When the CPU frequency of the MEC server is low, the waiting time and execution time of the task are longer, and the task completion rate of four algorithms is slightly lower. When the processing capacity of the MEC server is 7 GHz, all tasks can be completed within the deadline of the task. The task execution cost, which consists of task completion time and terminal energy consumption among four algorithms, tends to be the same when the processing capacity of the MEC server reaches 7 GHz.

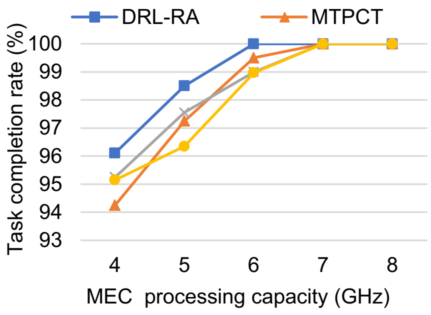
5.2.2.3 The influence of decision weight on algorithm performance Figure 11a–d, respectively, shows the influence of decision weight on algorithm performance. As the weight of the delay decision increases, the average service delay of the task gradually decreases. When delay decision weight was increased from 0.1 to 0.9, the average service delay of the task was shortened by 160 ms. The growth rate of the increase in terminal energy consumption slows as the weight of energy consumption decisions gradually decreases. The energy consumption of the terminal device will increase. When the energy consumption decision weight is reduced from 0.9 to 0.1, the terminal energy consumption of the task increases by 21.93 mJ. The task completion rate increases as the decision-making weight increases with time. DRL-RA needs to meet the needs of differ-



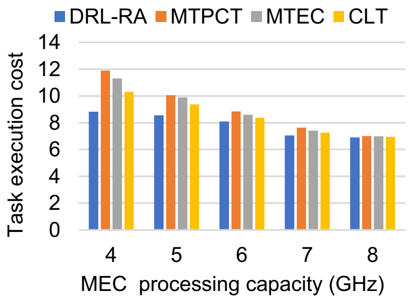
(a) Variation of average service delay with)
MEC server processing capacity



(b) Terminal energy consumption changes with
MEC server processing capacity

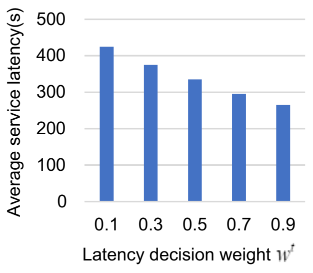


(c) The change of task completion rate with
MEC server processing capacity

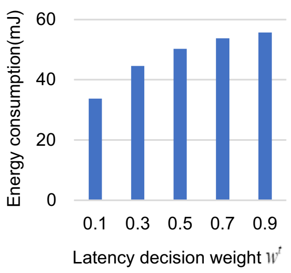


(d) the change of task execution cost with
MEC server processing capacity

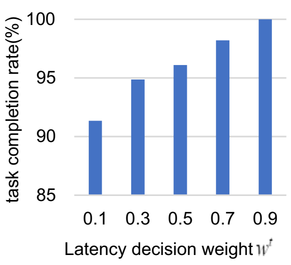
Fig. 10 The influence of the processing capacity of the MEC server on the experimental results



(a) Variation of average service delay
with decision weight



(b) Change of terminal energy
consumption with decision weight

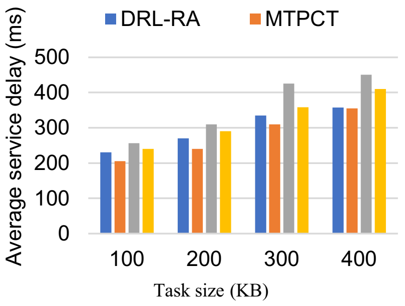


(c) The change of task completion
rate with decision weight

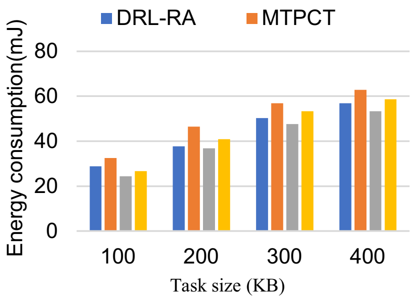
Fig. 11 The influence of decision weight on algorithm performance

ent users, choose different energy consumption and time decision weights for different tasks, and allocate communication and computing resources reasonably.

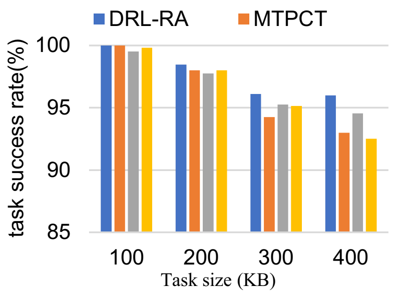
5.2.2.4 The effect of task size on the experimental results Figure 12a–d, respectively, show the average service delay, terminal energy consumption, task completion rate, task execution cost changes with the size of the computing task. When the task size is 400 kb, the difference between the average service delay and the MTPCT is very small. When the task size is 400 kb, DRL-RA is 20.67% lower than the MTEC and 12.93% lower than the CLT in terms of average service delay. When the task size is 400 kb, DRL-RA is 6.02 mJ less than MTPCT, 2 mJ less than CLT, and 3.5 mJ more than MTEC. DRL-RA uses DRL to reasonably allocate communication resources and computing resources, and the task success rate is higher than the other three algorithms. When the task size is 100 kb, the task execution time is short and the task success rate is higher. When the task size is 400 kb, DRL-RA increases the task success rate by 2.98% over MTPCT, 1.44% over MTEC, and 3.48% over CLT. When the task size is 400 kb, DRL-RA reduces the task execution cost by 22.98% compared to MTPCT, 21.75% lower than MTEC.



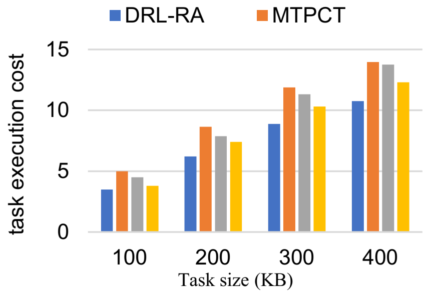
(a) Change of average service delay with task size



(b) Change of energy consumption with task size



(c) Change of task success rate with task size



(d) Change of task execution with task size

Fig. 12 The effect of task size on experimental results

6 Conclusion and future work

In the SDN-based MEC environment, to solve the frequent handover and ping-pong effect caused by the dense deployment of equipment, this paper comprehensively considers the AP traffic load and the MEC load, as well as the signal reception strength. The load-sensing based seamless switching method of wireless access point is proposed. Firstly, the system framework of multi-access edge computing environment based on SDN is introduced, and the seamless handoff problem is analyzed. Then, the uplink signal interference plus noise ratio, AP and edge server load model are integrated. Finally, a seamless handoff algorithm based on quantum particle swarm optimization is proposed to select AP to be accessed. Experimental results show that our scheme can reduce the average number of handovers and handover delay and improve system throughput and handover success rate and make the system load more balanced.

Because of the huge number of mobile devices and limited resources, the state of the wireless channel changes dynamically over time, this paper comprehensively considers the task completion time, local execution cost, and MEC server execution cost factors, which proposes a joint allocation of communication and computing resources based on deep reinforcement learning method. Firstly, the resource allocation problem is analyzed; then, the constraints of task completion time, local execution cost, MEC server execution cost and other factors are comprehensively considered; finally, the deep reinforcement learning is used to solve the task unloading vector and resource allocation vector to realize the joint allocation of communication resources and computing resources, so as to reduce the task execution cost. Experimental results show that this strategy has a higher task completion rate and lower task execution cost.

The seamless handover scheme in this paper does not involve the prediction of user movement. Therefore, the next step will be to build a user mobility model, predict the user's status and location, and integrate it into LASH to further improve the handover efficiency. The resource allocation scheme in this paper does not consider the energy consumption of the edge server during task processing and does not consider the impact of the delay and energy consumption during task return on the resource allocation scheme when the calculation result is large. At the same time, how to speed up the training speed of DRL is also the focus of the next step. In the future, more environmental factors and mobile device variables will be considered to make better migration decisions for mobile devices. It has better performance in delay and energy consumption. Mobile devices can adapt to the instability of energy supply and environmental network.

Acknowledgements The work was supported by the National Natural Science Foundation (NSF) under Grants (No. 61873341, 61771354), Key Research and Development Plan of Hubei Province (No. 2020BAB102), and Open Foundation of Industrial Software Engineering Technology Research and Development Center of Jiangsu Education Department (No. ZK19-04-04). Any opinions, findings, and conclusions are those of the authors and do not necessarily reflect the views of the above agencies.

References

1. Moazenzadeh R, Mohammadi B (2019) Assessment of bio-inspired metaheuristic optimisation algorithms for estimating soil temperature. Geoderma 353:152–171
2. Donepudi S, Garg S, Agarwal K, et al. (2014) Dynamic multi-access wireless network virtualization

3. Kim H, Feamster N (2013) Improving network management with software defined networking. IEEE Commun Mag 51(2):115–169
4. Baktir AC, Ozgovde A, Ersoy C (2017) How can edge computing benefit from software-defined networking: a survey, use cases, and future directions. IEEE Commun Surv Tutor 19(4):2359–2391
5. Li C, Song M, Zhang M, Luo Y (2020) Effective replica management for improving reliability and availability in edge-cloud computing environment. J Parallel Distrib Comput 143:107–128
6. Li C, Song M, Yu C, Luo YL (2021) Mobility and marginal gain based content caching and placement for cooperative edge-cloud computing. Inf Sci 548(16):153–176
7. Zhang K, Mao Y, Leng S et al (2017) Mobile-edge computing for vehicular networks: a promising network paradigm with predictive off-loading. IEEE Veh Technol Mag 12(2):36–44
8. Huang A, Nikaein N, Stenbock T, et al. (2017) Low latency MEC framework for SDN-based LTE/LTE-A networks. In: 2017 IEEE international conference on communications. Washington: IEEE Computer Society Press, pp 1–6
9. Katov AN, Mihovska A, Prasad NR (2015) Hybrid SDN architecture for resource consolidation in MPLS networks. In: Wireless telecommunications symposium. IEEE
10. Schiller E, Nikaein N, Kalogeiton E et al (2018) CDS-MEC: NFV/SDN-based application management for MEC in 5G systems. Comput Netw 135:96–107
11. Peng H, Ye Q, Shen XS (2019) SDN-based resource management for autonomous vehicular networks: a multi-access edge computing approach. IEEE Wirel Commun 26(4):156–162
12. Miladinovic I, Schefer-Wenzl S, Hirner H (2019) IoT architecture for smart cities leveraging machine learning and SDN. In: 2019 27th Telecommunications Forum. Washington: IEEE Computer Society Press, pp 1–4
13. Xia W, Zhang J, Quek TQS et al (2020) Mobile edge cloud-based industrial internet of things: improving edge intelligence with hierarchical SDN controllers. IEEE Veh Technol Mag 15(1):36–45
14. Wang J, Hu J, Min G et al (2019) Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning. IEEE Commun Mag 57(5):64–69
15. Tahaei H, Ko K, Seo W et al (2017) A QoE based trustable SDN framework for IoT devices in mobile edge computing. Springer, Singapore
16. Yazdinejad A, Parizi RM, Dehghantanha A et al (2020) An energy-efficient SDN controller architecture for IoT networks with blockchain-based security. IEEE Trans Serv Comput 13:625–638
17. Bao W, Yuan D, Yang Z et al (2017) Follow me fog: toward seamless handover timing schemes in a fog computing environment. IEEE Commun Mag 55(11):72–78
18. Yunoki K, Shinbo H (2018) Carry-on state service handover between edge hosts for latency strict applications in mobile networks. In: 2018 21st international symposium on wireless personal multimedia communications. Washington: IEEE Computer Society Press, pp 472–477
19. Neto AJV, Silva FSD, Neto EDP et al (2020) A taxonomy of DDoS attack mitigation approaches featured by SDN technologies in IoT scenarios. Sensors 20(11):3078
20. Bi Y, Han G, Lin C et al (2018) Mobility support for fog computing: an SDN approach. IEEE Commun Mag 56(5):53–59
21. Zeljković E, Slamnik-Kriještorac N, Latré S et al (2019) ABRAHAM: machine learning backed proactive handover algorithm using SDN. IEEE Trans Netw Serv Manage 16(4):1522–1536
22. Yin X, Wang L (2017) A fast handover scheme for SDN based vehicular network. In: International conference on mobile ad-hoc and sensor networks. Berlin: Springer, pp 293–302
23. Mouawad N, Naja R, Tohme S (2019) Fast and seamless handover in software defined vehicular networks. In: 2019 eleventh international conference on ubiquitous and future networks. Washington: IEEE Computer Society Press, pp 484–489
24. Zhang Y, Deng RH, Bertino E, et al. (2019) Robust and universal seamless handover authentication in 5G HetNets. In: IEEE transactions on dependable and secure computing, pp (99): 1–1
25. Mohseni H, Eslamnour B (2019) Handover management for delay-sensitive IoT services on wireless software-defined network platforms. In: 2019 3rd international conference on internet of things and applications. Washington: IEEE Computer Society Press, pp 1–6
26. Bi Y, Han G, Lin C et al (2019) Mobility management for intro/inter domain handover in software-defined networks. IEEE J Sel Areas Commun 37(8):1739–1754
27. Zhong X, Wang X, Li L, et al. (2020) A cooperative learning framework for resource management in MEC: an ADMM perspective
28. Lyu X, Tian H, Sengul C et al (2017) Multiuser joint task offloading and resource optimization in proximate clouds. IEEE Trans Veh Technol 66(4):1–1
29. Tran TX, Pompili D (2018) Joint task offloading and resource allocation for multi-server mobile-edge computing networks. IEEE Trans Veh Technol 68(1):856–868

30. Cheng K, Teng Y, Sun W, et al. (2018) Energy-efficient joint offloading and wireless resource allocation strategy in multi-MEC server systems. In: 2018 IEEE international conference on communications. Washington: IEEE Computer Society Press, pp 1–6
31. Liang C, He Y, Yu FR, et al. (2017) Energy-efficient resource allocation in software-defined mobile networks with mobile edge computing and caching. In: 2017 IEEE conference on computer communications workshops. Washington: IEEE Computer Society Press, pp 121–126
32. Wang P, Yao C, Zheng Z et al (2018) Joint task assignment, transmission, and computing resource allocation in multilayer mobile edge computing systems. IEEE Internet Things J 6(2):2872–2884
33. Qian LP, Shi B, Wu Y et al (2020) NOMA enabled mobile edge computing for internet of things via joint communication and computation resource allocations. IEEE Internet Things J 7(1):718–733
34. Yang Z, Liu Y, Chen Y, et al. (2019) Deep reinforcement learning in cache-aided MEC networks. In: ICC 2019–2019 IEEE international conference on communications. Washington: IEEE Computer Society Press, pp 1–6
35. Li C, Zhang Y, Zhiqiang H et al (2020) An effective scheduling strategy based on hypergraph partition in geographically distributed datacenters. Comput Networks 170:107096
36. Li C, Bai J, Yi C et al (2020) Resource and replica management strategy for optimizing financial cost and user experience in edge cloud computing system. Inf Sci 516:33–55
37. Li C, Tang J, Ma T, Yang X, Luo Y (2020) A workflow job scheduling algorithm based on load balancing in distributed cloud. J Network Comput Appl 152:1518
38. Wang S, Xu J, Zhang N et al (2018) A survey on service migration in mobile edge computing. IEEE Access 6:23511–23528
39. Han Z, Lei T, Lu Z et al (2019) Artificial intelligence-based handover management for dense WLANs: a deep reinforcement learning approach. IEEE Access 7:31688–31701
40. Banday Y, Rather GM, Begh GR (2019) SINR analysis and interference management of macrocell cellular networks in dense urban environments. Wirel Pers Commun 111:1–21
41. Guo S, Xiao B, Yang Y, et al. (2016) Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In: IEEE INFOCOM 2016-the 35th annual IEEE International conference on computer communications. Washington: IEEE Computer Society Press, pp 1–9
42. Chen L, Qu H, Zhao J et al (2016) Efficient and robust deep learning with correntropy-induced loss function. Neural Comput Appl 27(4):1019–1031
43. Liu W, Anguelov D, Erhan D, et al. (2016) Ssd: single shot multibox detector. European conference on computer vision. Berlin: Springer, pp 21–37
44. Chen J, Chen S, Wang Q et al (2019) iRAF: a deep reinforcement learning approach for collaborative mobile edge computing IoT networks. IEEE Internet Things J 6(4):7011–7024
45. Dai H, Zeng X, Yu Z et al (2019) A scheduling algorithm for autonomous driving tasks on mobile edge computing servers. J Syst Architect 94:14–23
46. Chen M, Hao Y (2018) Task offloading for mobile edge computing in software defined ultra-dense network. IEEE J Sel Areas Commun 36(3):587–597
47. Guo H, Zhang J, Liu J et al (2018) Energy-aware computation offloading and transmit power allocation in ultradense IoT networks. IEEE Internet Things J 6(3):4317–4329
48. Rajule N, Ambudkar B, Dhande A (2013) Survey of vertical handover decision algorithms. Int J Innov Eng Technol 2(1):362–368
49. Larasati HT, Hakimi R, Juhana T (2017) Extended-LLF: a least loaded first (LLF)-based handover association control for software-defined wireless network. Int J Comput Eng Inf Technol 9(9):203
50. Goutam S, Unnikrishnan S (2019) QoS based vertical handover decision algorithm using fuzzy logic. In: 2019 international conference on nascent technologies in engineering. Washington: IEEE Computer Society Press, pp 1–7
51. Kobayashi R, Adachi K (2019) Radio and computing resource allocation for minimizing total processing completion time in mobile edge computing. IEEE Access 7:141119–141132
52. Nguyen PD, Ha VN, Le LB (2019) Computation offloading and resource allocation for backhaul limited cooperative MEC systems. In: 2019 IEEE 90th vehicular technology conference. Washington: IEEE Computer Society Press, pp 1–6

Chunlin Li is a Professor of Computer Science in Wuhan University of Technology. She received the ME in Computer Science from Wuhan Transportation University in 2000 and PhD in Computer Software and Theory from Huazhong University of Science and Technology in 2003. Her research interests include cloud computing and distributed computing.

Yong Zhang received his MS degree from Tiangong University in 2020. He is currently working toward the PhD degree with the School of Computer Science and Technology, Wuhan University of Technology. His research interests include cloud computing and artificial intelligence.

Youlong Luo He is a Vice-Professor of Management at Wuhan University of Technology. He received his MS in Telecommunication and System from Wuhan University of Technology in 2003 and his PhD in Finance from Wuhan University of Technology in 2012. His research interests include cloud computing and electronic commerce.