Cluster Computing (2023) 26:2897–2911
https://doi.org/10.1007/s10586-022-03768-z

DRJOA: intelligent resource management optimization through deep reinforcement learning approach in edge computing

Yifan Chen[1,2] · Shaomiao Chen[3] · Kuan-Ching Li[4] · Wei Liang[3] · Zhiyong Li[1,2]

Received: 20 May 2022 / Revised: 20 September 2022 / Accepted: 26 September 2022 / Published online: 15 December 2022

Abstract
Mobile edge computing (MEC) can enhance the computation capabilities of smart mobile devices for computation-intensive mobile applications via supporting computation offloading efficiently. However, the limitation of wireless resources and computational resources of edge servers often becomes the bottlenecks to realizing the developments of MEC. In order to address the computation offloading problem in the time-varying wireless networks, the offloading decisions and the allocation of radio and computation resources need to be jointly managed. Traditional optimization methods are challenging to deal with the combinatorial optimization problem in complex real-time dynamic network environments. Therefore, we propose a deep reinforcement learning (DRL)-based optimization approach, named DRJOA, which jointly optimizes offloading decisions, computation, and wireless resources allocation. The optimization algorithm based on DRL has the advantages of fast solving speed and strong generalization ability, which makes it possible to solve combinatorial optimization problems online. Simulation results show that our proposed DRJOA in this study dramatically outperforms the benchmark methods for offloading decisions and system utility.

Keywords Computation offloading · Deep reinforcement learning · Mobile edge computing · Resource allocation.

1 Introduction

In recent years, the computation capability and energy efficiency of mobile devices have become extremely important with the emergence of new mobile applications (e.g., augmented reality and 3D games) [1–3]. Mobile edge computing (MEC) can effectively relieve the pressure of increasing computing demands on mobile devices, reduce task execution latency, and extend the battery life of terminals by offloading the computationally intensive tasks to neighboring edge servers [4–6]. However, although task offloading to edge servers can effectively utilize computational resources at the server side, the experience quality of users varies significantly under different task offloading strategies and network resource allocation states. The reason for this is that both the computation resources of the edge server and the radio resources in the network are limited. The computation offloading strategy is not always practical, and a poor computation offloading decision may even worsen task processing delay and increase energy consumption. Therefore, designing a reasonable task offloading strategy and allocating computational and radio

✉ Wei Liang
wliang@hnust.edu.cn

✉ Zhiyong Li
zhiyong.li@hnu.edu.cn

Yifan Chen
cyf176@hnu.edu.cn

Shaomiao Chen
csm123@hnust.edu.cn

Kuan-Ching Li
kuancli@pu.edu.tw

1 College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China

2 Key Laboratory for Embedded and Network Computing of Hunan Province, Changsha 410082, China

3 School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, China

4 Department of Computer Science and Information Engineering, Providence University, Taichung 43301, Taiwan

resources in MEC systems are vital challenges that must be addressed [7–11].

Many studies deal with optimization problems related to computation offloading decisions and resource allocation in MEC systems [7, 12–17]. Considering the presence of binary offloading variables in these problems, they are typically categorized as mixed integer nonlinear programming (MINP) problems. Currently, exact approaches such as branch-and-bound algorithms [18] and dynamic programming [19] can be utilized to solve MINP problems; however, when the problem scales up, these algorithms will consume massive computation time and are difficult to scale. In addition, there are approximate approaches, such as heuristic local search [13, 20, 21] and relaxation algorithms [22, 23]. Compared to the exact approaches, the approximate approaches can search for a solution within an acceptable computation time. However, when the problem becomes complicated, a large number of iterations still lead to a considerable computation time. Hence, approximation methods are difficult to be applied to online, real-time computation offloading optimization problems.

In recent years, with the development of artificial intelligence technology, deep learning has been widely used in various fields [24, 25]. In this paper, to solve task offloading decisions and resource allocation in MEC systems online, we propose a deep reinforcement learning (DRL)-based offloading framework that works as an adaptive scheduler to learn binary offloading decisions. Edge servers can provide computing services to users, reducing task execution latency and energy consumption of mobile devices. Therefore, we use processing latency and energy consumption as evaluation metrics to analyze the performance of an MEC system. We define the user's utility function as the weighted sum of the improvement ratios on delay and energy consumption via edge computing compared with local computing. Our study aims to maximize the system utility, which is defined as the weighted sum of the utility functions of all users in the system. The significant contributions of this study are as follows.

1. We construct a joint optimization problem of computation offloading, communication, and computational resource allocation in a time-varying MEC network to maximize system utility, which is a mixed integer nonlinear programming (MINP) problem.
2. We propose a DRL-based joint optimization approach (DRJOA) for each user in a time-varying MEC system to obtain an optimal offloading strategy and the corresponding resource allocation. DRJOA converts the original optimization problem into two subproblems: wireless and computational resource allocation and computation offloading decision problems. The proposed DRJOA learns the offloading generation strategy from past offloading experiences under the different wireless channels. The computation offloading decision problem uses a deep neural network (DNN) and K nearest neighbor (KNN) methods to generate offloading decisions efficiently. Thus, DRJOA selects the best offloading decision from a limited decision space, and its computational complexity does not grow dramatically with the size of the problem.
3. We conduct extensive simulations to evaluate the performance of our proposed DRJOA algorithm. The simulation results demonstrate that the system utility of the proposed solution is near-optimal, and the proposed DRJOA algorithm significantly improves system offloading utility compared with benchmark approaches.

The remainder of this paper is organized as follows. The related work is introduced in Sect. 2. Sections 3 and 4 describe the system model and problem formulation, respectively. Section 5 elaborates on the working principle of DRJOA to solve the optimal resource allocation and computation offloading problems. Finally, the experimental results are analyzed in Sect. 6, and conclusions are provided in Sect. 7.

2 Related work

In MEC networks, jointly optimizing computation offloading decisions and resource allocation have been extensively studied in academia and industry [14]. Many related studies [12, 13, 22, 23, 26, 27] have defined it as an MINP problem. For example, [26] presents a branch-and-bound method based on a reconstruction linearization technique to obtain optimal or suboptimal results by setting the accuracy of the solution. Considering that the complexity of this method was not guaranteed, the authors further designed a greedy heuristic algorithm to solve the MINP problem by reducing it into a convex problem. In [12], the authors decomposed the original MINP problem in a single-cell multi-user scenario into a resource allocation subproblem and a task offloading decision subproblem. The closed-form solution of the resource allocation subproblem was obtained via convex and quasi-convex optimization techniques, transforming the MINP problem into an integer programming problem; the offloading solution was obtained via a heuristic algorithm. In [13], the authors studied a joint optimization problem of resource allocation and computational offloading in a multi-cell multi-user system and solved the problem using a heuristic algorithm. In [27], the authors focused on optimizing the offloading decisions of all users, computing access points, and communication and computation resource allocation to minimize the weighted sum of energy costs, computation, and maximum delay among all users. They proposed a heuristic algorithm based on semi-definite relaxation and a novel randomization mapping method to solve the problem. However, applying these algorithms to real-time MEC systems still faces many challenges. The primary reason is that MEC systems are highly real-time systems wherein the channel changes rapidly. The heuristic algorithms mentioned above require numerous iterations to obtain the local optimum; thus, they do not meet real-time requirements.

To solve this problem, we introduce the DRL algorithm in the MEC system, which can use DNNs to learn the best mapping of state space to action space. At present, studies have been conducted on computation offloading methods based on DRL in MEC systems [28–33]. In order to minimize the overall cost in terms of time and energy for all users, a scheme based on Deep Q-Networks (DQN) is proposed in [28] to optimize computation offloading and resource allocation jointly. Without prior knowledge of the network dynamics, [29] presents a computation offloading algorithm based on a dual DQN that learns the optimal strategy. To address the problem of DQN-based methods for selecting actions by exhaustive search, [30] proposed a new DRL framework that maximizes the weighted sum of computation rates in wirelessly powered MEC networks. Specifically, in each iteration, only a tiny fraction of candidate offloading actions are considered in [30], leading to near-optimal offloading actions. Based on the advantages and disadvantages of the prior studies, the current work considers a DRL-based scheme with an action space that contains just a limited quantity of offloading decisions to solve the joint optimization problem of computation offloading and wireless and computational resource allocation in single-cell MEC networks.

3 System model

In this section, we introduce the system model. For easy reading and understanding, some notations frequently used in this paper and their corresponding descriptions are listed in Table 1.

We consider a single-cell MEC system, as shown in Fig. 1. The system is composed of a base station s and multiple mobile users $\mathcal{N} = \{1, \ldots, N\}$, and users communicate with the base station through wireless channels. The base station is equipped with an edge server, which uses its powerful computing resources to provide computing services to mobile users. Mobile devices have different computing capabilities due to various limitations, such as physical size, battery capacity, and functionality. Each mobile device is assumed to have computational tasks that must be processed, and these tasks can be handled locally by the device or offloaded to the edge server.

3.1 Task model

We assume that each user D_i has a computationally intensive task $\mathcal{J}_i \triangleq (d_i, w_i)$ to be processed. Here, d_i is the size of the input data of task $\mathcal{J}_i$, and w_i indicates the number of CPU cycles required to complete the task. Information about d_i and w_i can be obtained using the program analysis methods presented in [34, 35].

Tasks are supposed to be indivisible and cannot be decomposed into multiple subtasks. When user D_i has a task to perform, it first sends a message to the base station containing information about that task. Then, it waits for an offloading decision from the base station. There are two offloading decisions in total, one is offloading to the edge server, and the other is local processing. Due to their varying computing capabilities, the energy and time required to process the same task differ between mobile devices and the edge server. The base station considers the computation task requirements of all users and ultimately determines the offloading policies for all users. The offloading decision of user D_i is denoted as $a_i \in \{0, 1\}$. If the base station determines that the computational tasks of user D_i are processed locally at the user's device, then $a_i = 0$; conversely, $a_i = 1$.

3.2 Communication model

The base station s manages users' uplink and downlink communications in the network. Mobile devices are assumed to access the network via orthogonal frequency division multiplexing. The channel is divided into several orthogonal sub-channels for communication between a mobile device and the base station s. Various devices occupy different communication sub-channels for their respective transmission to eliminate intra-cell interference [36]. If the offloading decision of user D_i is $a_i = 1$, then the uplink rate $r_i(p_i)$ at which user D_i transmits data at power p_i can be expressed as

$$r_i(p_i) = B \log_2\left(1 + \frac{p_i g_i}{B N_0}\right), \tag{1}$$

where B is the bandwidth for each orthogonal sub-channel, g_i represents the wireless channel gain of user D_i, and N_0 denotes the Gaussian noise power spectral density at the receiver of base station s. The total system bandwidth is W. Therefore, the total number of mobile users in the network that can simultaneously access base station s cannot exceed $M = \lfloor W/B \rfloor$. In accordance with (1), mobile user D_i can

Table 1 Description of Key Notations

Notation	Description
$\mathcal{N}$, N	Set/number of mobile device users
$\mathcal{J}_i$	Computation task of mobile device user D_i
d_i	Input data size of task $\mathcal{J}_i$
w_i	CPU cycles required for processing computation task $\mathcal{J}_i$
c_i^l	Local computation capacity of user D_i
c_s	Computation capacity of the edge server
ρ_i	Percentage of computing resources allocated to user D_i
t_i^l	Local execution time of computation task $\mathcal{J}_i$
t_i^{es}	Delay for mobile user D_i in mobile edge cloud computing
$t_i^{\mathrm{es,t}}$	Transmission delay of mobile user D_i for offloading task $\mathcal{J}_i$
$t_i^{\mathrm{es,e}}$	Processing delay of task $\mathcal{J}_i$ at the MEC server
e_i^{l}	Energy consumption for mobile user D_i in local computing
e_i^{es}	Energy consumption for user D_i in edge computing
ε_i	Energy consumption coefficient
r_i	Transmission rate of user D_i for computation offloading
W	System bandwidth
B	User bandwidth
p_i	Transmission power of user D_i
$p_{\max}$	Maximum permissible transmission power
g_i	Channel gain between BS s and user D_i
N_0	Gaussian noise power spectral
a_i	$a_i = 1$ if task $\mathcal{J}_i$ is offloaded; otherwise, $a_i = 0$
$\boldsymbol{a}$	$= (a_1, a_2, \ldots, a_I)$; decision profile of all users
λ_i^t	Weights on computation time
λ_i^e	Weights on energy consumption
$\mathcal{N}_s$	Set of offloading users

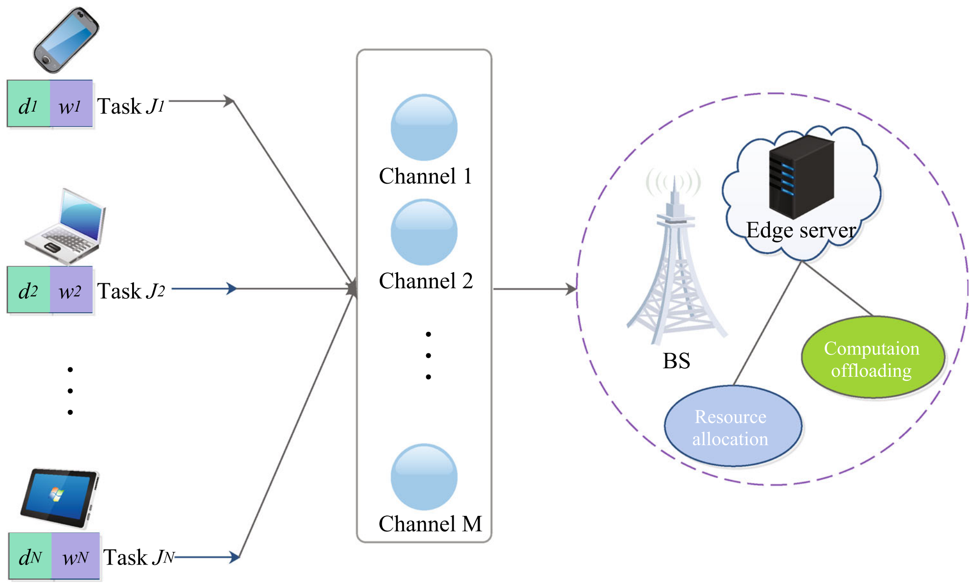

Fig. 1 A single-cell multi-user MEC system

change the uplink transmission rate $r_i(p_i)$ by adjusting its upload power p_i. Assuming that the maximum value of the upload power p_i is $p_{\max}$, the uplink transmission rate $r_i(p_i)$ varies within the range of $(0, B\log_2(1+p_{\max}g_i/BN_0)]$.

3.3 Computation model

This section analyzes the computational overhead of completing a task using both local and edge computing approaches. The computational overhead is mainly related to the time and energy consumed to complete the task.

3.3.1 Local computing

For the local computing method, mobile user D_i processes its task $\mathcal{J}_i$ by using its computational resources. Let c_i^l be the local computation capability of mobile user D_i. Then, the time required to process the task locally for user D_i is

$$t_i^l = \frac{w_i}{c_i^l}. \tag{2}$$

The energy consumption of user D_i is

$$e_i^l = \varepsilon_i w_i, \tag{3}$$

where ε_i is the energy consumption factor that represents the energy consumed per CPU cycle. In accordance with [37, 38], ε_i is a superlinear function of computation capability and denoted as

$$\varepsilon_i = z \cdot \left(c_i^l\right)^{\beta}, \tag{4}$$

where z denotes the parameter related to the device chip architecture, and β is typically set to 2.

3.3.2 Edge computing

The tasks are processed on the edge servers for the edge computing method. The complete offloading process consists of the following three parts.

- Device D_i sends task-related data to the base station over a wireless channel. Then, the base station forwards this information to the edge server.
- The edge server allocates appropriate computation resources to execute the task.
- After the task is processed, the output is returned to device D_i.

Similar to [12, 39], for many applications (e.g., face recognition), the output data size of the task is much smaller than the input data size, so we ignore the overhead incurred by transmitting the output data back to the user.

The transmission time $t_i^{es,t}(p_i)$ and energy consumption $e_i^{es}(p_i)$ of user D_i for offloading input data d_i to the edge server can be computed as

$$t_i^{\mathrm{es,t}}(p_i) = \frac{d_i}{r_i(p_i)} = \frac{d_i}{B\log_2(1+p_i g_i/BN_0)} \tag{5}$$

and

$$\begin{aligned} e_i^{es}(p_i) =& p_i \cdot t_i^{\mathrm{es,t}}(p_i) = p_i \cdot \frac{d_i}{r_i(p_i)} \\ =& p_i \cdot \frac{d_i}{B\log_2(1+p_i g_i/BN_0)}. \end{aligned} \tag{6}$$

After the task $\mathcal{J}_i$ of device D_i is offloaded to the edge server, the edge server will handle this task. Let c_s represent the computational capacity of the edge server and ρ_i be a percentage indicating the computational resources assigned to task $\mathcal{J}_i$ by edge server s. The processing time $t_i^{es,e}(\rho_i)$ for task $\mathcal{J}_i$ of device D_i by edge computing is given by:

$$t_i^{\mathrm{es,e}}(\rho_i) = \frac{w_i}{\rho_i c_s}. \tag{7}$$

Hence, the total time overhead $t_i^{es}(p_i,\rho_i)$ of processing task $\mathcal{J}_i$ using edge computing method consists of transmission time and edge processing time, which can be expressed as

$$\begin{aligned} t_i^{es}(p_i,\rho_i) =& t_i^{\mathrm{es,t}}(p_i) + t_i^{\mathrm{es,e}}(\rho_i) \\ =& \frac{d_i}{B\log_2(1+p_i g_i/BN_0)} + \frac{w_i}{\rho_i c_s}. \end{aligned} \tag{8}$$

4 Problem formulation

In an MEC system, regardless of whether the task is processed locally ($a_i = 0$) or at the edge server ($a_i = 1$), the execution delay of task $\mathcal{J}_i$ can be written as

$$\begin{aligned} t_i =& (1-a_i)t_i^l + a_i t_i^{es} \\ =& (1-a_i)\frac{w_i}{c_i^l} + a_i\left(\frac{d_i}{r_i(p_i)} + \frac{w_i}{\rho_i c_s}\right). \end{aligned} \tag{9}$$

Similarly, the energy consumption for task $\mathcal{J}_i$ can be expressed as

$$\begin{aligned} e_i =& (1-a_i)e_i^l + a_i e_i^{es} \\ =& (1-a_i)\varepsilon_i w_i + a_i p_i \cdot \frac{d_i}{r_i(p_i)}. \end{aligned} \tag{10}$$

We define the utility of user D_i, $u_i(a_i, p_i, \rho_i)$ as

$$u_i(a_i, p_i, \rho_i) = \lambda_i^t \frac{t_i^l - t_i}{t_i^l} + \lambda_i^e \frac{e_i^l - e_i}{e_i^l} = a_i\left(\lambda_i^t \frac{t_i^l - t_i^{es}}{t_i^l} + \lambda_i^e \frac{e_i^l - e_i^{es}}{e_i^l}\right), \tag{11}$$

where $\lambda_i^e \in [0,1]$ and $\lambda_i^t \in [0,1]$ denote mobile device user D_i's preferences for energy consumption and computation time, respectively. Considering that the time and energy metrics differ, we need to perform a normalization operation. Furthermore, since we need to compare the performance of local and edge computing, we use $(t_i^l - t_i)/t_i^l$ and $(e_i^l - e_i)/e_i^l$ to eliminate the effect of different metrics. $(t_i^l - t_i)/t_i^l$ and $(e_i^l - e_i)/e_i^l$ represent the performance improvement in the time and energy consumption of task $\mathcal{J}_i$ executed via edge computing compared with that by local computing. The higher the utility function of mobile device user D_i is, the lower the cost of completing task $\mathcal{J}_i$ is; moreover, the better service user D_i receives from the MEC system. To analyze the computing services provided by the system to all mobile device users, we define the system utility as

$$R(\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho}) = \sum_{i=1}^{N} \kappa_i u_i(a_i, p_i, \rho_i) = \sum_{i=1}^{N} \kappa_i a_i \left(\lambda_i^t \frac{t_i^l - t_i^{es}}{t_i^l} + \lambda_i^e \frac{e_i^l - e_i^{es}}{e_i^l}\right), \tag{12}$$

where κ_i denotes the system preference for mobile device user D_i. κ_i can be set based on user type and the urgency of the computing task (e.g., tasks of first responders and police officers should be set higher priority with a high κ_i). In addition, κ_i is related to the price users pay for computing services.

All users' offloading decisions and allocating communication and computational resources to the users for the MEC system are defined as a joint optimization problem. The optimization goal is to maximize the system utility so that the computing resources of the edge server can be fully utilized. Taking the constraints of offloading decision, transmission power, server computing capability, and wireless channel into consideration, we formulate this optimization problem as follows:

$$\begin{aligned} \max_{\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho}} \quad & R(\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho}) \\ s.t. \quad C_1 & : a_i \in \{0,1\}, \forall i \in \mathcal{N}, \\ C_2 & : 0 < p_i \le p_{\max}, \forall i \in \mathcal{N}_s, \\ C_3 & : \rho_i > 0, \forall i \in \mathcal{N}_s, \\ C_4 & : \textstyle\sum_{i \in \mathcal{N}_s} \rho_i \le 1, \\ C_5 & : \textstyle\sum_{i \in \mathcal{N}} a_i \le M, \end{aligned} \tag{13}$$

where $\boldsymbol{a}$, $\mathbf{p}$, and $\boldsymbol{\rho}$ denote the computation offloading decision vector, transmission power vector, and resource allocation percentage vector of all users, respectively. $\mathcal{N}_s$ is the set of users who choose edge computing.

Constraint C_1 states that each user can choose only one between local computing or edge computing. Constraint C_2 specifies the valid transmission power range for each user. Constraint C_3 ensures that the edge server must allocate computing resources to the users who choose edge computing. Constraint C_4 ensures that the total computing resources provided by the edge server to the users who choose edge computing cannot exceed its maximum available computing resources. Constraint C_5 guarantees that the total number of users choosing edge computing simultaneously cannot be larger than the maximum number of subchannels M.

5 DRL-based approach: DRJOA

Since there are three variables that need to be determined in Problem (13) defined above, namely, $\boldsymbol{a}$, $\mathbf{p}$ and $\boldsymbol{\rho}$, where $\boldsymbol{a}$ is a binary vector, $\mathbf{p}$ and $\boldsymbol{\rho}$ are continuous vectors. Besides, the objective function $R(\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho})$ in (13) is nonlinear. Therefore, Problem (13) is a MINP problem, which is NP-hard. It is known that there is no efficient way to solve NP-hard problems. To solve Problem (13), we first rewrite Problem (13) as P:

$$\begin{aligned} P : & \max_{\boldsymbol{a}} \max_{\mathbf{p}, \boldsymbol{\rho}} R(\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho}) \\ & s.t. \quad C_1, C_2, C_3, C_4, \text{ and } C_5. \end{aligned} \tag{14}$$

Suppose that offloading decision $\boldsymbol{a}$ is fixed, Problem P can be rewritten as:

$$\begin{aligned} P_1 : & \max_{\mathbf{p}, \boldsymbol{\rho}} R(\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho}) \\ & s.t. \quad C_2, C_3, \text{ and } C_4. \end{aligned} \tag{15}$$

Hence, we decompose Problem P into two subproblems.

1. Optimal resource allocation subproblem. This subproblem is Subproblem P_1, which involves optimization variables $\mathbf{p}$ and $\boldsymbol{\rho}$ and can be solved by convex and quasiconvex optimization methods [12].
2. Computation offloading subproblem. The computational offloading subproblem is a combinatorial optimization problem. Due to the fast-changing wireless channel, a fast policy response is required, and the traditional numerical optimization methods are difficult to achieve online offloading decisions. Thus, we use a DRL-based method to address the online optimization problem of computation offloading.

5.1 Optimal resource allocation subproblem

Substituting (12) into (15), the objective function of P_1 can be written as

$$\max_{\boldsymbol{p},\boldsymbol{\rho}} \sum_{i\in\mathcal{N}_s} \kappa_i\left(\lambda_i^t+\lambda_i^e\right)-\sum_{i\in\mathcal{N}_s} \kappa_i\left(\lambda_i^t t_i^{es}/t_i^l+\lambda_i^e e_i^{es}/e_i^l\right). \tag{16}$$

The first term of (16) is constant for a particular offloading decision and independent of the resource allocation variables $\mathbf{p}$ and ρ. Therefore, we can transform objective function (16) into

$$\min_{\boldsymbol{p},\boldsymbol{\rho}} \sum_{i\in\mathcal{N}_s} \kappa_i\left(\lambda_i^t t_i^{es}/t_i^l+\lambda_i^e e_i^{es}/e_i^l\right)$$

.

Based on the above analysis and (1)-(8), we can transform Problem P1 into its equivalent problem as follows:

$$\begin{aligned} P_1^{'}: \min_{\mathbf{p},\boldsymbol{\rho}} \quad & \sum_{i\in\mathcal{N}_S} \frac{\varphi_i c_i^l}{\rho_i}+\frac{\delta_i+\xi_i p_i}{\log_2(1+p_i g_i/BN_0)} \\ s.t. \quad & C_2, C_3, \text{ and } C_4, \end{aligned} \tag{17}$$

where $\varphi_i=\kappa_i\lambda_i^t/c_s$, $\delta_i=\kappa_i\lambda_i^t d_i/\left(Bt_i^l\right)$, and $\xi_i=\kappa_i\lambda_i^e d_i/\left(Be_i^l\right)$. Following Problem $P_1^{'}$, we notice that the power assignment p_i and computational resource allocation ρ_i are mutually independent in terms of both the objective function and the constraints. We transform the original Problem $P_1^{'}$ into two subproblems: optimal computational resource allocation and transmission power allocation. In the following, we solve these two subproblems separately.

5.1.1 Computational resource allocation

Using the first term of the objective function in (17) as the new objective function, the computational resource allocation problem is decoupled from Problem $P_1^{'}$ and can be expressed as:

$$\begin{aligned} P_2: \min_{\boldsymbol{\rho}} \quad & \sum_{i\in\mathcal{N}_S} \frac{\varphi_i c_i^l}{\rho_i} \\ s.t. \quad & C_3 \text{ and } C_4. \end{aligned} \tag{18}$$

Let $f(\boldsymbol{\rho})=\sum_{i\in\mathcal{N}_S}\varphi_i c_i^l/\rho_i$. Each term in the Hessian matrix of $f(\boldsymbol{\rho})$ can be written as

$$\frac{\partial^2 f}{\partial\rho_i\partial\rho_j}=\begin{cases} 2\varphi_i c_i^l/\rho_i^3, & \text{if } i=j, \\ 0, & \text{otherwise} . \end{cases} \tag{19}$$

From (19), we observe that all the diagonal elements of the Hessian matrix are positive, and the other elements are zero. Thus, the Hessian matrix of $f(\boldsymbol{\rho})$ is positive definite. In addition, the domain of $f(\boldsymbol{\rho})$ is convex. Therefore, $f(\boldsymbol{\rho})$ is a convex function.

Problem P_2 can be addressed by applying Lagrange multiplier approaches. We define the Lagrange function $L(\boldsymbol{\rho},\mu,\tau)$ related to Problem P_2 as

$$L(\boldsymbol{\rho},\mu,\tau)=\sum_{i\in\mathcal{N}_S}\frac{\varphi_i c_i^l}{\rho_i}+\mu\left(\sum_{i\in\mathcal{N}_S}\rho_i-1\right)-\sum_{i\in\mathcal{N}_S}\rho_i\tau_i, \tag{20}$$

where μ and τ_i are the Lagrange multipliers for Constraints C_4 and C_3. Based on the Karush-Kuhn-Tucker conditions, the optimal ρ_i^*, μ^*, and τ_i^* need to satisfy the conditions below:

$$\begin{cases} \dfrac{\partial L}{\partial\rho_i^*}=-\dfrac{\varphi_i c_i^l}{\rho_i^{*2}}+\mu^*-\tau_i^*=0, \forall i\in\mathcal{N}_S; \\ \mu^*\left(\sum_{i\in\mathcal{N}_s}\rho_i^*-1\right)=0; \\ \rho_i^*\tau_i^*=0, i\in\mathcal{N}_S; \\ \tau_i^*\geq 0, \rho_i^*>0, i\in\mathcal{N}_S; \mu\geq 0, \end{cases} \tag{21}$$

By solving (21), we derive an optimal ρ_i^* in closed form, and it is formulated as

$$\rho_i^*=\frac{\sqrt{\varphi_i c_i^l}}{\sum_{i\in\mathcal{N}_s}\sqrt{\varphi_i c_i{}^l}}. \tag{22}$$

5.1.2 Transmission power allocation

The transmission power allocation problem is defined by

$$\begin{aligned} P_3: \min_{\mathbf{p}} \quad & \sum_{i\in\mathcal{N}_s}\frac{\delta_i+\xi_i p_i}{\log_2(1+p_i g_i/BN_0)} \\ s.t. \quad & C_2: 0<p_i\leq p_{\max}, \forall i\in\mathcal{N}_S. \end{aligned} \tag{23}$$

Each term in the objective function of Problem P_3 (i.e., $\frac{\delta_i+\xi_i p_i}{\log_2(1+p_i g_i/BN_0)}$) is positive and independent of one another. Thus, Problem P_3 can be transformed into a set of Problem P_4, which is defined as follows:

$$\begin{aligned} P_4: \min_{p_i} \quad & \frac{\delta_i+\xi_i p_i}{\log_2(1+p_i g_i/BN_0)} \\ s.t. \quad & C_2 \quad : 0<p_i\leq p_{\max}, \forall i\in\mathcal{N}_S. \end{aligned} \tag{24}$$

Problem P_4 is quasi-convex and can be addressed utilizing a bisection method [12].

5.2 Computation offloading subproblem

Based on the analysis in the previous sections, we know that given an offloading decision $\boldsymbol{a}$, the optimal uplink power $\mathbf{p}^*$ and computational resource allocation $\boldsymbol{\rho}^*$ can be obtained. We also get the optimal system utility with respect to the offloading decision $\boldsymbol{a}$, denoted as

$$\begin{aligned}\mathcal{F}^*(\boldsymbol{a}) &= \max_{\boldsymbol{p},\boldsymbol{\rho}} R(\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho}) \\ &= \sum_{i\in\mathcal{N}_s} \kappa_i\left(\lambda_i^t + \lambda_i^e\right) - h(\boldsymbol{\rho}^*) - \sum_{i\in\mathcal{N}_s} g(p_i^*),\end{aligned} \tag{25}$$

where $h(\boldsymbol{\rho}^*) = \sum_{i\in\mathcal{N}_S} \frac{\varphi_i c_i^l}{\rho_i^*}$ can be calculated by substituting (22) into (18), and $g(p_i^*) = \frac{\delta_i + \xi_i p_i^*}{\log_2\left(1+p_i^* g_i/BN_0\right)}$. Subsequently, we need to search for the optimal computation offloading decision via solving the following problem:

$$\begin{aligned}\max_{\boldsymbol{a}} & \sum_{i\in\mathcal{N}_s} \kappa_i\left(\lambda_i^t + \lambda_i^e\right) - h(\boldsymbol{\rho}^*) - \sum_{i\in\mathcal{N}_s} g(p_i^*) \\ s.t. \quad & C_1, \text{ and } C_5.\end{aligned} \tag{26}$$

Problem (26) is a combinatorial optimization problem. We use a DRL-based method to solve Problem (26) to obtain real-time offloading decisions.

Combined with the above analysis on the resource allocation subproblem, we propose a DRL-based online offloading optimization framework called DRJOA, as shown in Fig. 2. In the framework, DRJOA comprises four modules: DNN, KNN, offloading decision selection (ODS), and data caching module. The proposed DRJOA aims to approximate a function h that yields an output $\{\boldsymbol{a}^*, \boldsymbol{p}^*, \boldsymbol{\rho}^*\}$ when the system takes the channel gain $\boldsymbol{g}$ as the input. System utility $R^*(\boldsymbol{a}^*, \mathbf{p}^*, \boldsymbol{\rho}^*)$ acts as the reward. For simplicity, function h can be given by:

$$h : \boldsymbol{g} \mapsto \boldsymbol{a}^*. \tag{27}$$

In the following, we will introduce the DNN, KNN, ODS, and data caching modules.

5.2.1 DNN module

DNN has a powerful feature representation capability. Taking advantage of this feature, DRJOA adopts DNN to adapt to offloading decisions. DNN in DRJOA is used to generate an N-dimensional vector $\boldsymbol{a}_c^{'}$, which can be represented by function h_1, as follows:

$$h_1 : \boldsymbol{g} \mapsto \boldsymbol{a}_c^{'}. \tag{28}$$

The input of this DNN is a vector $\boldsymbol{g}$ composed of the channel gains of N users, and its output is an N-dimensional vector $\boldsymbol{a}_c^{'}$. Besides, there are also two hidden layers in the DNN module. Therefore, the number of both input and output neurons is N. For the two hidden layers, the number of neurons is specified as 120 and 80, and the rectified linear unit (ReLU) is employed as the activation function, wherein the relationship between the output y of a neuron and its input x is $y = \max\{x, 0\}$. For the output layer, we use the sigmoid function $y = 1/(1 + e^{-x})$ as the activation function, such that each element $a_{c,i}{'}$ in the output vector $\boldsymbol{a}_c^{'}$ satisfies $a_{c,i}{'} \in (0, 1)$.

5.2.2 KNN module

We use the KNN method to transform $\boldsymbol{a}_c^{'}$ into K binary offloading decisions. The function of the KNN module can be represented in mathematical terms as

$$h_2 : \boldsymbol{a}_c^{'} \mapsto \{\boldsymbol{a}_k \mid \boldsymbol{a}_k \in \{0, 1\}^N, k = 1, \cdots, K\}, \tag{29}$$

where K is an integer within the interval $[1, 2^N]$. With larger K, the higher the computational complexity of the DRJOA algorithm, the closer the solution is to the optimal one, and vice versa. Here, we set the value of K to N.

In DRJOA, KNN selects K nearest binary vectors to vector $\boldsymbol{a}_c^{'}$ by calculating the distance between $\boldsymbol{a}_c^{'}$ and each binary vector from the 2^N N-dimensional binary vectors. A simple example is provided below to help understand how the KNN module works. Assume that $\boldsymbol{a}_c^{'} = [0.1, 0.2, 0.3,$

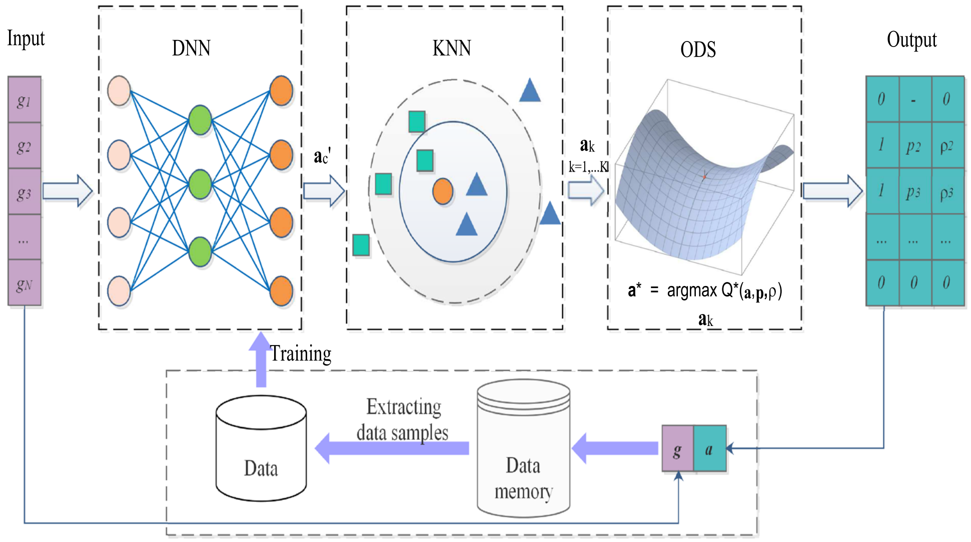

Fig. 2 DRJOA framework

0.4] and $K = 4$. The distance between binary vectors [0, 0, 0, 0], [0, 0, 0, 1], ..., [1, 1, 1, 1] and vector $\boldsymbol{a}_c^{'}$ ($\boldsymbol{a}_c^{'} = [0.1, 0.2, 0.3, 0.4]$) is computed. The distances are 0.5477, 0.7071, 0.8367, 0.9487, 0.9487, 1.0488, 1.1402, 1.2247, 1.0488, 1.1402, 1.2247, 1.3038, 1.3038, 1.3784, 1.4491, and 1.5166. Therefore, the four nearest vectors to vector $\boldsymbol{a}_c^{'}$ are $\boldsymbol{a}_1 = [0, 0, 0, 0]$, $\boldsymbol{a}_2 = [0, 0, 0, 1]$, $\boldsymbol{a}_3 = [0, 0, 1, 0]$, and $\boldsymbol{a}_4 = [0, 0, 1, 1]$.

5.2.3 ODS module

In the ODS module, the optimal resource allocation $\mathbf{p}$, $\boldsymbol{\rho}$ for each offloading decision is analyzed in detail above, and the corresponding reward $R^*(\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho})$ can be computed. Then, the offloading decision that corresponds to the maximum value from these rewards is selected as the optimal offloading decision $\boldsymbol{a}^*$ when the input is $\mathbf{g}$, that is, $\boldsymbol{a}^* = \mathrm{argmax} R(\boldsymbol{a}_k, \mathbf{p}^*, \boldsymbol{\rho}^*)$. Finally, the optimal offloading decision $\boldsymbol{a}^*$ and the corresponding resource allocation policies $\mathbf{p}^*$, $\boldsymbol{\rho}^*$ obtained via the ODS module constitute the output of the system. The optimal resource allocation scheme is described in detail in Sect. 5.1.

5.2.4 Data caching module

The data is collected to train the DNN module to enable DRJOA to make better offloading decisions. The network stores the input $\mathbf{g}$ and the output offloading decision $\boldsymbol{a}^*$ for each time slot t as data samples in the data memory. Suppose the data memory size is L. If the memory is complete, the freshly obtained state-action pair $\{\mathbf{g}, \boldsymbol{a}^*\}$ will be added to replace the oldest stored data. A batch of data is used to train the DNN by extracting data from the data storage every Δ time slot. During the training process, we use the Adam algorithm [40] to update the parameters of the DNN to decrease the average cross-entropy loss, as

$$L(\theta_t) = -\frac{1}{|\mathcal{S}_t|}\sum_{\tau\in\mathcal{S}_t}\Big(\big(\mathbf{a}_\tau^*\big)^\top \log f_{\theta_t}(\mathbf{g}_\tau) + \big(1-\mathbf{a}_\tau^*\big)^\top \log(1-f_{\theta_t}(\mathbf{g}_\tau))\Big),$$

Algorithm 1 DRJOA algorithm

```
Input: Wireless channel gain $\mathbf{g}_t$ at time frame $t$
Output: Offloading decision $\boldsymbol{a}_t^*$ and the corresponding resource allocation policies $\mathbf{p}_t^*$, $\boldsymbol{\rho}_t^*$
1:  Initialization:
    Initialize the DNN with random parameters $\theta_1$
    Empty the data memory
2:  for $t = 1, 2, ..., T$ do
3:      Generate an N-dimensional vector $\boldsymbol{a}_c^{'}$ from the DNN
4:      Transform $\boldsymbol{a}_c^{'}$ into $K$ binary offloading decisions $\{\boldsymbol{a}_k, k = 1, \cdots, K\}$ by KNN
5:      Compute the system utility $R^*(\boldsymbol{a}_k, \mathbf{p}^*, \boldsymbol{\rho}^*)$ at each $\mathbf{a}_k$ by solving Sub-problem $P_1$
6:      Select $\boldsymbol{a}_t^* = \arg\max_{\{\mathbf{a}_k\}} R(\boldsymbol{a}_k, \mathbf{p}^*, \boldsymbol{\rho}^*)$
7:      Store $(\mathbf{g}_t, \boldsymbol{a}_t^*)$ into the data memory
8:      if $t\%\Delta == 0$ then
9:          Extract a batch of data $\mathcal{S}_t$ from the data storage
10:         Train the DNN with $\mathcal{S}_t$ and update parameters $\theta_t$ by the Adam algorithm
11:     end if
12: end for
```

where $\mathcal{S}_t$ is the training data set extracted from the data storage, and $|\mathcal{S}_t|$ is the number of samples in the set. The Adam algorithm is an improved type of gradient descent method that adaptively adjusts the learning rate of gradient descent. Thus, it can converge more efficiently and avoids manual adjustment of the learning rate.

The proposed DRJOA algorithm is provided in Algorithm 1. Initially, the parameters θ_1 in the DNN are set randomly, and the data memory is empty. Then, the algorithm goes through the KNN, ODS, and data caching modules. The DNN parameters are continuously improved through learning, thus improving the offloading strategy.

6 Simulations

In this section, to analyze the performance of DRJOA, we compare DRJOA with the following benchmark schemes.

• Local computing (LC): The tasks of all mobile device users are processed locally on their own devices.

• Enumeration (Enum): We enumerate all task offloading policies to find the global optimal solution.

• All edge computing (AEC): The offload decisions for all users are $a_i = 1, \forall i \in \mathcal{N}$. However, considering the limited bandwidth of the system, users who do not have acquired bandwidth resources can only handle their tasks on their devices.

• Random computing (RC): Offloading decisions are made randomly.

6.1 Experiment setup

All simulation experiments are performed in a network with a hexagonal region whose side length is 288 m. In the center of the region, there is a base station equipped with an edge server. Users are randomly distributed in the network. For wireless communication, the system bandwidth is assumed to be $W = 8$ MHz, and user bandwidth is set to $B = 1$ MHz. As a result, at most eight users are allowed to send data simultaneously. According to the wireless channel interference model in [41], the channel gain is $g_i = d_{i,s}^{-\alpha}$, where $d_{i,s}$ denotes the distance between user D_i and base station s, and α denotes the path loss exponent. During the simulation, we consider face detection and recognition as tasks. The size of the input data is $d_i = 420$ KB. The total amount of CPU cycles required to process this computational task is $w_i = 1000$ MCycles. The parameters employed in the simulation are summarized, as shown in Table 2.

The proposed DRJOA algorithm is implemented in python based on TensorFlow. In the DRJOA algorithm, we use a fully connected DNN with one input layer, two hidden layers, and one output layer, where the number of neurons in two hidden layers is 120 and 80, respectively. We set the training interval to 10, the batch size of training data to 128, the size of data memory L = 1024, and the learning rate of the Adam optimizer to 0.01.

6.2 Convergence performance

We assume that the system time is divided into several consecutive time frames of equal length, which are set to be less than the channel coherence time. Also, the channel is assumed to remain unchanged within a time frame but may differ in different time frames. Figure 3 shows the variation of the training loss with the time frame for the DRJOA algorithm. The DRJOA algorithm has a much higher training loss in the initial training phase than in the later training phase. In order to be able to show the variation of training loss in one figure, we replace the training loss greater than 1 with 1 in the plotting process. During the initial phase of training, the training loss fluctuates significantly. As the number of iterations increases, the training loss gradually decreases. Consequently, the training loss curve tends to be smooth, and the model converges to the optimal offloading strategy.

Figure 4 shows how the normalized utility changes as the time frame increases. The normalized utility $R_n(\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho})$ is defined as

$$R_n(\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho}) = \frac{R^*(\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho})}{\max_{\boldsymbol{a}' \in \{0,1\}^N} R^*(\boldsymbol{a}', \mathbf{p}, \boldsymbol{\rho})}, \tag{30}$$

where the denominator in (30) denotes the optimal utility obtained by finding the optimal policy by searching the entire offloading space and computing the corresponding optimal resource allocation policy, the numerator denotes the utility obtained by the DRJOA algorithm. Evidently, $R_n(\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho}) \in [0, 1]$; and the larger the value of $R_n(\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho})$ is, the closer the policy obtained by the DRJOA algorithm is to the optimal one.

In Fig. 4, the blue curves indicate the average of the normalized utility $R_n(\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho})$ over the last 50 time frames, while the light blue shade indicates the maximum and minimum values of $R_n(\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho})$ over the last 50 time frames. As the time frame increases, the normalized utility $R_n(\boldsymbol{a}, \mathbf{p}, \boldsymbol{\rho})$ obtained by the DRJOA algorithm gradually converges to the optimal solution.

6.3 Offloading decision

In this subsection, we evaluate the offloading decision accuracy of different algorithms through simulation experiments. Offloading decision accuracy is measured as

Table 2 Simulation data parameter

Parameter	Numerical value	Unit
N	10	–
W	8	MHz
B	1	MHz
z, β	$z = 5 \times 10^{-27}$, $\beta = 2$	–
α	4	–
$p$max	1	w
N_0	10^{-16}	–
d_i	420	KB
w_i	1000	MCycles
c_i^l	0.5-1.5	GHz
c_s	20	GHz
λ_i^t, λ_i^e	0.5, 0.5	–
κ_i	1	–

the ratio of correctly predicted samples to the total amount of samples tested. Obviously, the higher the ratio, the more accurate the algorithm prediction is and the closer it is to the system's best performance.

Figure 5 shows the offloading decision accuracy obtained by different offloading schemes. Accuracy varies widely among different methods. Thus, we present two different scales, as shown in Fig. 5. We observe that the left Y-axis in Fig. 5 indicates an accuracy interval from 0 to 1.1, and the right Y-axis indicates an accuracy interval from 0 to 0.005, with a minimum scale of 0.0005. The offloading policy obtained by the Enum algorithm is optimal; hence, accuracy is set to 1. The offloading decision accuracy of algorithms LC, RC, AEC, and our proposed DRJOA is 0.00, 0.0011, 0.0016, and 0.8870, respectively. The offloading decision accuracy of our proposed DRJOA algorithm outperforms those of other offloading schemes.

6.4 System utility

Figure 6 compares the average normalized utility obtained by different offloading schemes. The average normalized utility results from averaging the normalized utility obtained during the testing phase. Given that the Enum algorithm obtains the optimal utility, the average normalized utility value is set as 1. From Fig. 6, the average normalized utility obtained by algorithms LC, RC, AEC, and our proposed DRJOA are 0.00, 0.2988, 0.2479, and 0.9800, respectively. The average normalized utility of our proposed DRJOA is considerably better than those of the other methods.

7 Conclusion

This study proposes a DRL-based online optimization approach DRJOA for computation offloading and resource allocation in single-cell networks. In particular, we first introduce the system model, which includes the task,

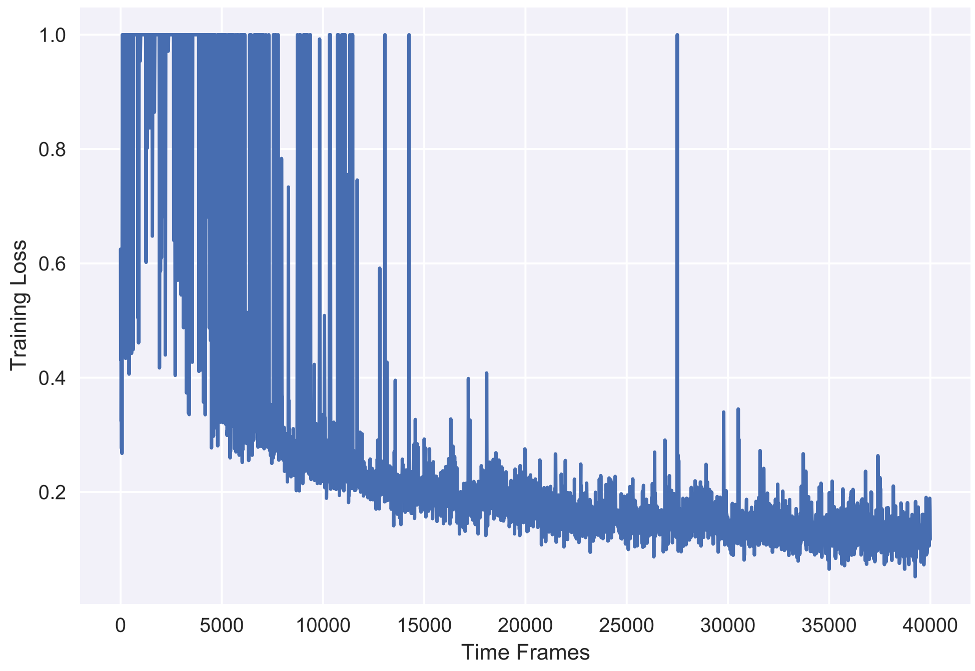

Fig. 3 Training loss of the DRJOA algorithm varies with the time frame

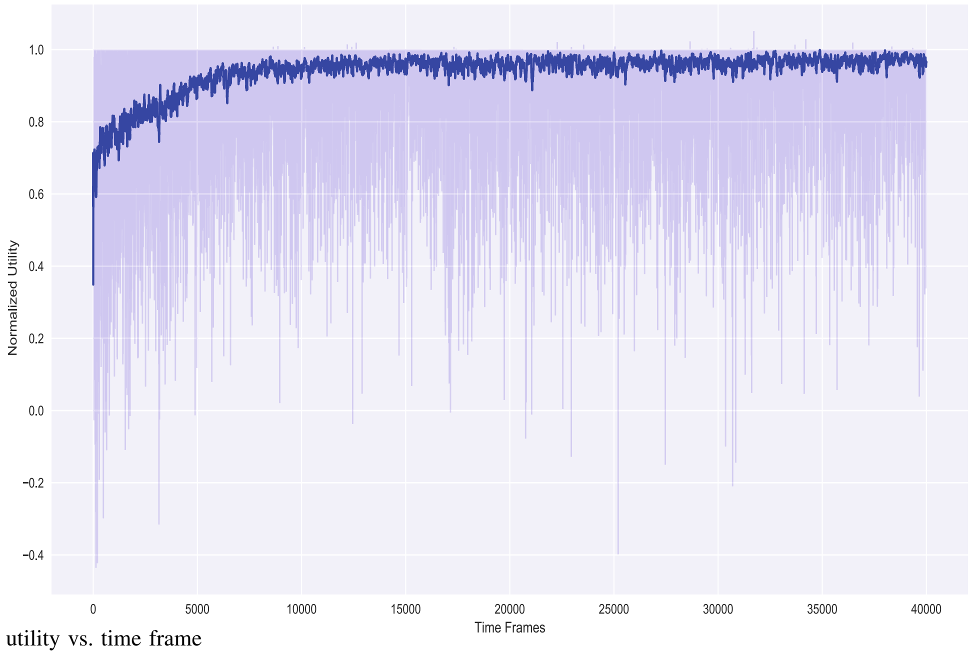

Fig. 4 Normalized utility vs. time frame

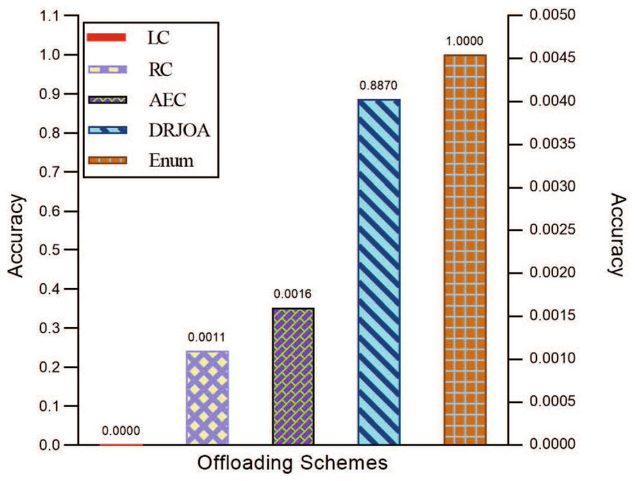

Fig. 5 Comparison of offloading decision accuracy

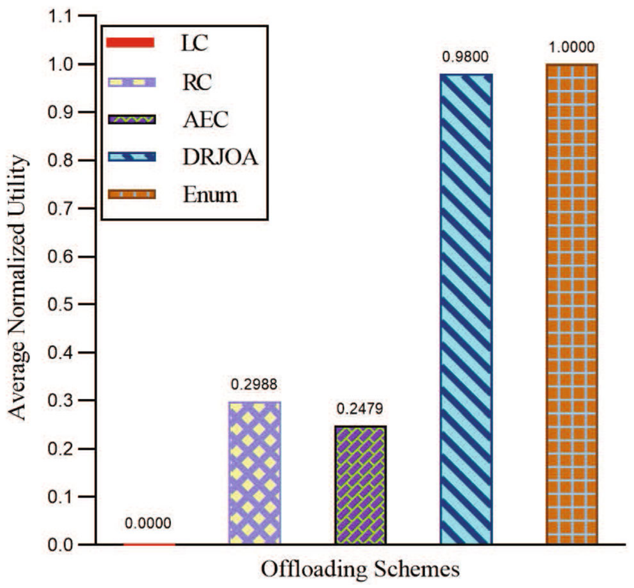

Fig. 6 Comparison of the average normalized utility obtained using different offloading schemes

communication, and computation models. Then, we establish the optimization problem to maximize system utility by jointly considering computation offloading and wireless and computational resource allocation. After that, we propose a DRL-based algorithm, i.e., DRJOA, to obtain a near-optimal solution rapidly for the optimization problem. In DRJOA, we use convex and quasi-convex optimization techniques to solve the wireless and computational resource allocation problem and construct a DRL method to handle the computation offloading problem. The results confirm that DRJOA outperforms other benchmark algorithms under the evaluation metrics of offloading decision accuracy and system utility. The results also demonstrate that the algorithm converges.

Author contributions YC conceived the presented idea. YC and SC designed the model and the computational framework, and analyzed the problem models and design algorithms. YC conducted simulation experiments and analyzed the results. YC and SC wrote the

manuscript with input from all authors. KL, WL and ZL conceived the study and were in charge of overall direction and planning.

Funding This work was partially supported by National Key Research and Development Program of China (No. 2018YFB1308604), National Natural Science Foundation of China (Nos. U21A20518, 61976086, 61906065), State Grid Science and Technology Project (No. 5100-202123009A), Special Project of Foshan Science and Technology Innovation Team (No. FS0AA-KJ919-4402-0069), and Hunan Natural Science Foundation (No. 2020JJ5200).

Data availability The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declared that they do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

Human and animal rights This work did not include humans or animal participants.

References

1. Li, K.: A game theoretic approach to computation offloading strategy optimization for non-cooperative users in mobile edge computing, IEEE Trans. Sustain. Comput. pp. 1–1 (2018)
2. Xu, X., Zhang, X., Gao, H., Xue, Y., Qi, L., Dou, W.: Become: blockchain-enabled computation offloading for iot in mobile edge computing. IEEE Trans. Ind. Inform. **16**(6), 4187–4195 (2020)
3. Arthur Sandor, V.. K., Lin, Y., Li, X., Lin, F., Zhang, S.: Efficient decentralized multi-authority attribute based encryption for mobile cloud data storage. J. Netw. Comput. Appl. **129**, 25–36 (2019)
4. Long, C., Cao, Y., Jiang, T., Zhang, Q.: Edge computing framework for cooperative video processing in multimedia IOT systems. IEEE Trans. Multimed. **20**, 1126–1139 (2018)
5. Liu, C., Li, K., Liang, J., Li, K.: COOPER-MATCH: Job offloading with a cooperative game for guaranteeing strict deadlines in mec. IEEE Trans. Mobile Comput. pp. 1–1 (2019)
6. Yi, C., Cai, J., Su, Z.: A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications. IEEE Trans. Mobile Comput. **19**(1), 29–43 (2020)
7. Wang, C., Liang, C., Yu, F.R., Chen, Q., Tang, L.: Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. IEEE Trans. Wirel. Commun. **16**(8), 4924–4938 (2017)
8. Chen, M., Hao, Y.: Task offloading for mobile edge computing in software defined ultra-dense network. IEEE J. Sel. Areas Commun. **36**(3), 587–597 (2018)
9. Zhao, J., Li, Q., Gong, Y., Zhang, K.: Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks. IEEE Trans. Veh. Technol. **68**(8), 7944–7956 (2019)
10. Zhou, W., Chen, L., Tang, S., Lai, L., Xia, J., Zhou, F., Fan, L.: Offloading strategy with PSO for mobile edge computing based on cache mechanism. Clust. Comput. **25**(4), 2389–2401 (2022)
11. Bacanin, N., Antonijevic, M., Bezdan, T., Zivkovic, M., Venkatachalam, K., Malebary, S.: Energy efficient offloading mechanism using particle swarm optimization in 5g enabled edge nodes, Clust. Comput. pp. 1–12 (2022)
12. Lyu, X., Tian, H., Sengul, C., Zhang, P.: Multiuser joint task offloading and resource optimization in proximate clouds. IEEE Trans. Veh. Technol. **66**, 3435–3447 (2017)
13. Tran, T.X., Pompili, D.: Joint task offloading and resource allocation for multi-server mobile-edge computing networks. IEEE Trans. Veh. Technol. **68**, 856–868 (2019)
14. Du, J., Yu, F.R., Chu, X., Feng, J., Lu, G.: Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization. IEEE Trans. Veh. Technol. **68**(2), 1079–1092 (2019)
15. Li, H., Xu, H., Zhou, C., Lü, X., Han, Z.: Joint optimization strategy of computation offloading and resource allocation in multi-access edge computing environment. IEEE Trans. Veh. Technol. **69**(9), 10214–10226 (2020)
16. Zhang, D., Tang, J., Du, W., Ren, J., Yu, G.: Joint optimization of computation offloading and ul, dl resource allocation in mec systems. In: IEEE 29th annual international symposium on personal. Indoor Mobile Radio Commun. (PIMRC), pp. 1–6 (2018)
17. Huang, P.-Q., Wang, Y., Wang, K., Liu, Z.-Z.: A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing. IEEE Trans. Cybern. **50**(10), 4228–4241 (2020)
18. Narendra, P., Fukunaga, K.: A branch and bound algorithm for feature subset selection. IEEE Trans. Comput. **26**, 917–922 (1977)
19. Bertsekas, D.: Dynamic programming and optimal control (1995)
20. Bi, S., Zhang, Y.: Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. IEEE Trans. Wirel. Commun. **17**, 4177–4190 (2018)
21. Li, Z., Chen, S., Zhang, S., Jiang, S., Gu, Y., Nouioua, M.: FSB-EA: fuzzy search bias guided constraint handling technique for evolutionary algorithm. Expert Syst. Appl. **119**, 20–35 (2019)
22. Guo, S., Xiao, B., Yang, Y., Yang, Y.: Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In: IEEE INFOCOM 2016—the 35th Annual IEEE International Conference on Computer Communications, pp. 1–9 (2016)
23. Dinh, T.Q., Tang, J., La, Q., Quek, T.Q.S.: Offloading in mobile edge computing: task allocation and computational frequency scaling. IEEE Trans. Commun. **65**, 3571–3584 (2017)
24. Liang, W., Li, Y., Xie, K., Zhang, D., Li, K.-C., Souri, A., Li, K.: Spatial-temporal aware inductive graph neural network for c-its data recovery. In: IEEE Transactions on Intelligent Transportation Systems, pp. 1–12 (2022)
25. Diao, C., Zhang, D., Liang, W., Li, K.-C., Hong, Y., Gaudiot, J.-L.: A novel spatial-temporal multi-scale alignment graph neural network security model for vehicles prediction. In: IEEE Transactions on Intelligent Transportation Systems, pp. 1–11 (2022)
26. Zhao, P., Tian, H., Qin, C., Nie, G.: Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing. IEEE Access **5**, 11255–11268 (2017)
27. Chen, M.-H., Dong, M., Liang, B.: Joint offloading decision and resource allocation for mobile cloud with computing access point. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3516–3520 (2016)
28. Li, J., Gao, H., Lv, T., Lu, Y.: Deep reinforcement learning based computation offloading and resource allocation for mec. In: 2018 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–6 (2018)
29. Chen, X., Zhang, H., Wu, C., Mao, S., Ji, Y., Bennis, M.: Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning. IEEE Internet Things J. **6**, 4005–4018 (2019)
30. Huang, L., Bi, S., Zhang, Y.: Deep reinforcement learning for online computation offloading in wireless powered mobile-edge

computing networks. IEEE Trans. Mobile Comput. **19**, 2581–2593 (2020)
31. Zhan, Y., Guo, S., Li, P., Zhang, J.: A deep reinforcement learning based offloading game in edge computing. IEEE Trans. Comput. **69**, 883–893 (2020)
32. Du, J., Yu, F.R., Lu, G., Wang, J., Jiang, J., Chu, X.: MEC-assisted immersive VR video streaming over terahertz wireless networks: A deep reinforcement learning approach. IEEE Internet Things J. **7**(10), 9517–9529 (2020)
33. Mustafa, E., Shuja, J., Bilal, K., Mustafa, S., Maqsood, T., Rehman, F. et al.: Reinforcement learning for intelligent online computation offloading in wireless powered edge networks. Clust. Comput. pp. 1–10 (2022)
34. Cuervo, E., Balasubramanian, A., ki Cho, D., Wolman, A., Saroiu, S., Chandra, R., Bahl, P.:MAUI: making smartphones last longer with code offload, in: MobiSys '10, (2010)
35. Yang, L., Cao, J., Yuan, Y., Li, T., Han, A., Chan, A.: A framework for partitioning and execution of data stream applications in mobile cloud computing. In: 2012 IEEE Fifth International Conference on Cloud Computing pp. 794–802 (2012)
36. Sesia, S., Toufik, I., Baker, M.: LTE-the UMTS long term evolution: From theory to practice. (2011)
37. Wen, Y., Zhang, W., Luo, H.: Energy-optimal mobile application execution: taming resource-poor mobile devices with cloud clones. In: 2012 Proceedings IEEE INFOCOM. pp. 2716–2720 (2012)
38. Miettinen, A. P. , Nurminen, J.: Energy efficiency of mobile clients in cloud computing. In: HotCloud (2010)
39. Chen, X.: Decentralized computation offloading game for mobile cloud computing. IEEE Trans. Parall. Distrib. Syst. **26**, 974–983 (2015)
40. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations
41. Yang, L., Zhang, H., Li, M., Guo, J., Ji, H.: Mobile edge computing empowered energy efficient task offloading in 5G. IEEE Trans. Veh. Technol. **67**(7), 6398–6409 (2018)

Yifan Chen received the M.Sc. degree in the College of Computer Science and Electronic Engineering from Hunan University, Changsha, China, in 2015. She is currently pursuing the Ph.D. degree with the College of Computer Science and Electronic Engineering, Hunan University, China. Her current research interests include Internet of Things, distributed computing, game theory, and mobile edge computing.

Shaomiao Chen received the MSc and Doctoral degree in Computer Science and Technology from Hunan University, Changsha, China, in 2014 and 2018. He is currently a lecturer at School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China. His research interests include Intelligent Computing and Big Data Computing Systems.

Kuan-Ching Li is currently appointed as Distinguished Professor at Providence University, Taiwan. He is a recipient of awards and funding support from several agencies and high-tech companies, as also received distinguished chair professorships from universities in several countries. He has been actively involved in many major conferences and workshops in program/general/steering conference chairman positions and as a program committee member, and has organized numerous conferences related to high performance computing and computational science and engineering. Besides publication of journal and conference papers, he is the co-author/co-editor of several technical professional books published by CRC Press, Springer, McGraw-Hill, and IGI Global. His topics of interest include parallel and distributed computing, Big Data, and emerging technologies. He is a Member of the AAAS, a Senior Member of the IEEE, and a Fellow of the IET.

Wei Liang is currently a Professor at the School of Computer science and Engineering, Hunan University of Science and Technology, China. He received his Ph.D. degree at Hunan University in 2013 and was a postdoctoral scholar at Lehigh University, USA, during 2014-2016. He served as Application Track Chair of IEEE Trustcom 2015, a Workshop Chair of IEEE Trustcom WSN 2015 and IEEE Trustcom WSN 2016. He has published more than 110 journal/conference papers such as IEEE Transactions on Industrial Informatics, IEEE Transactions on Emerging Topics in Computing, and IEEE Internet of Things Journal. His research interests include Blockchain security technology, Networks Security Protection, embedded system and Hardware IP protection, Fog computing, and Security management in WSN. He is a Member of the IEEE.

Zhiyong Li (M'14) received the M.Sc. degree in system engineering from the National University of Defense Technology, Changsha, China, in 1996, and the Ph.D. degree in control theory and control engineering from Hunan University, Changsha, in 2004. Since 2004, he has been with the College of Computer Science and Electronic Engineering, Hunan University, where he is currently a Full Professor. He has authored over 100 papers in international journals and conferences. His research interests include intelligent perception and autonomous moving body, machine learning and industrial big data, and intelligent optimization algorithms with applications. He is a member of the China Computer Federation and the Chinese Association for Artificial Intelligence.