Wireless Networks (2021) 27:2023–2038
https://doi.org/10.1007/s11276-021-02554-w

Joint computation offloading and task caching for multi-user and multi-task MEC systems: reinforcement learning-based algorithms

Ibrahim A. Elgendy[1,2] · Wei-Zhe Zhang[1,3] · Hui He[1] · Brij B. Gupta[4,5] · Ahmed A. Abd El-Latif[6]

Accepted: 18 January 2021 / Published online: 9 February 2021

Abstract
Computation offloading at mobile edge computing (MEC) servers can mitigate the resource limitation and reduce the communication latency for mobile devices. Thereby, in this study, we proposed an offloading model for a multi-user MEC system with multi-task. In addition, a new caching concept is introduced for the computation tasks, where the application program and related code for the completed tasks are cached at the edge server. Furthermore, an efficient model of task offloading and caching integration is formulated as a nonlinear problem whose goal is to reduce the total overhead of time and energy. However, solving these types of problems is computationally prohibitive, especially for large-scale of mobile users. Thus, an equivalent form of reinforcement learning is created where the state spaces are defined based on all possible solutions and the actions are defined on the basis of movement between the different states. Afterwards, two effective Q-learning and Deep-Q-Network-based algorithms are proposed to derive the near-optimal solution for this problem. Finally, experimental evaluations verify that our proposed model can substantially minimize the mobile devices' overhead by deploying computation offloading and task caching strategy reasonably.

Keywords Computation offloading · Task caching · Energy-efficient · Mobile edge computing · Q learning · Deep Q Network

1 Introduction

Nowadays, mobile devices and wireless sensors support a wide variety of different applications and services such as augmented reality, face detection and recognition, video games, e-Health and natural language processing which typically demand more energy and computation processing for execution. However, limited resources and battery capacity pose a significant challenge and restrict these devices from executing such resource-demanding applications [2, 5, 35, 49, 55]. Mobile cloud computing is one of the prominent approaches which can overcome such

✉ Ibrahim A. Elgendy
ibrahim.elgendy@hit.edu.cn

Wei-Zhe Zhang
wzzhang@hit.edu.cn

Hui He
hehui@hit.edu.cn

Brij B. Gupta
gupta.brij@gmail.com

Ahmed A. Abd El-Latif
a.rahiem@gmail.com

[1] School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China

[2] Department of Computer Science, Faculty of Computers and Information, Menoufia University, Shebin El-Koom 32511, Egypt

[3] Peng Cheng Laboratory, Cyberspace Security Research Center, Shenzhen, China

[4] National Institute of Technology Kurukshetra, Kurukshetra, India

[5] Asia University, Taichung City, Taiwan

[6] Mathematics and Computer Science Department, Faculty of Science, Menoufia University, P.O. Box 32511, Shebin El-Koom, Egypt

limitations and expand the battery-life of these devices by offloading the intensive computations to a centralized cloud computing through a wireless channel [10, 20, 23, 57]. Although mobile cloud computing has many benefits, it also poses a range of different problems, such as high data latency and substantial overhead traffic to mobile devices that make delay-sensitive and real-time applications are not suitable for executing [28, 29].

Simultaneously, with the increase of multimedia services, the demand for the mobile cellular network is rising inexorably, which has posed a great challenge on the network capacity and backhaul links [3, 22, 51]. To address these issues and to decrease the communication delay, a new paradigm is known as mobile edge computing (MEC) introduced, in which cloud resources and services are moved to edge nodes (base stations) which are nearest to mobile devices and becomes a cost-efficient solution with low latency [39]. In addition, caching and delivery techniques are considered as a good solution to address the challenges of multimedia services, where duplicated transmissions can be avoided by caching popular contents at the network edge [52, 56]. With the development of MEC system's computation offloading, there are a lot of models and approaches have been recently conducted with different objectives related to energy consumption, resource allocation and delay requirements [19, 30, 39, 59]. In addition, caching and delivery techniques are applied on the contents and computation tasks to reduce the transmitted data over the cellular network. However, on the one hand, most of these developments are mainly concerned with the content caching, exploring an optimum caching policy, whereas a few studies deal with the task caching in which considering the computation offloading and task caching has a great effect on reducing the mobile devices' overhead. In the other hand, it is a challenge to implement an optimum approach in dynamic and complex systems like the multi-user MEC. Therefore, in this study, we first introduced the task caching concept where the application code and related library for the completed tasks are cached at the edge server. In addition, the task offloading and caching are jointly formulated as an efficient problem whose goal is to reduce the total overhead of mobile users. Finally, two efficient Q learning and Deep-Q-Network-based algorithms have been proposed to derive the near-optimal decision for this problem. The distinct contributions of our paper include:

- The caching concept, where the application program and related code for the completed tasks are cached at the edge server, is applied on the computation task to minimize delay and energy overhead. In addition, the computation capacity and task popularity are utilized to determine the task caching policy.
- An efficient model of task offloading and caching is formulated as a nonlinear problem whose goal is to reduce the time and energy cost of mobile users. This type of problem is NP-hard.
- An equivalent form of reinforcement learning is given from the above problem, in which the state spaces are defined based on all possible solutions and the actions are defined on the basis of movement between the different states. Then, two efficient Q Learning and Deep-Q-Network-based algorithms have been proposed to derive the near-optimum solution.
- Finally, simulations have been conducted to prove that the overall cost for the mobile users can be minimized significantly by deploying offloading and caching policy reasonably.

The remainder of this paper is organized as follows. Section 2 reviews the common policies in computation offloading. Section 3 presents our proposed model with the formulation of our problem. The proposed algorithms based on Q-learning and Deep-Q-Network are presented in Sect. 4. Experimental evaluation are provided in Sect. 5. Finally, the conclusion is deduced in Sect. 6.

2 Related work

Recently, various models have been proposed for computation offloading of multi-user MEC system. Most of these approaches have used traditional optimization methods such as convex optimization techniques and Lyapunov optimization for solving these problems [39]. Whereas latterly, Deep Learning algorithms have been introduced as powerful methods to solve these problems in an efficient way [7, 31, 54]. This section introduces a brief overview of the popular models on the basis of approaches used for solving these problems.

2.1 Traditional optimization methods

Minimizing the energy consumption under the communication delay constraint for MEC systems with multi-user is considered as the main goal of [13, 16, 40]. Mao et al. have formulated an optimization model which has jointly considered the radio and computation resources [40]. In addition, an algorithm is developed to allocate the bandwidth and transmission power for mobile users based on Lyapunov optimization. Whereas Chen et al. proposed an algorithm that jointly allocates the resources of computation and communication between device users and finds an optimum offloading decision through a separable semidefinite relaxation approach [13]. While in [16], the frequency scaling of the CPU-cycle and the task allocation

of mobile users have been jointly investigated for energy efficiency of MEC system. The authors formulated a stochastic problem where its main goal is to reduce the overhead of energy while guaranteeing the queuing length for the computation task. In addition, an efficient task offloading and frequency scaling algorithm, which solves this problem without the need for statistical information, is developed. However, the user computation tasks deadline requirement was neglected in these studies.

Moreover, some outstanding works have been addressed the content caching with the computation offloading to decrease the communication delay and improve the quality of users' services. For example, in [37], the content caching placement, content delivery and the request dispatch are jointly optimized in which an online learning approach is introduced to tackle the challenges of wireless networks with the edge-cloud system. However, Wang et al. have jointly formulated data caching and computation offloading as a problem whose objective is to reduce the resource-constrained delay [50]. In addition, a new heuristic algorithm is developed for solving this problem in an efficient way. However, caching the computation is not considered in these works while it has a significant effect on reducing the overall mobile devices' overhead. In addition, the characteristics of computation tasks are not considered in allocating the resources.

Recently, task caching has been considered with computation offloading for MEC systems in which the completed tasks and their related code are cached at the edge server for the next execution. Nur et al. [42] have jointly considered the caching scheme and offloading for the computation task as an optimization problem. In addition, the task priority is considered where it can be calculated based on data size, task popularity, deadline and computing resource. Furthermore, efficient algorithmic steps are developed to provide the solution for this problem. Whereas, Liu et al. [34] have focused on the communication resources by proposing a cache-enhanced MEC system for multiuser in which the duplicated data can be avoided from transmission in this system. In addition, an optimization problem of communication, computation resources and caching policy is formulated which aims to decrease the energy overhead whereas satisfying the communication latency constraint. However, in [34], the authors supposed that all the tasks will be cached at the edge server whereas, it's storage capacity is bounded and all the computation tasks cannot be cached.

2.2 Deep learning methods

Deep Learning algorithms are currently and commonly utilized in a broad range of fields, such as gaming, computer vision, robotics, network functions, internet of things [4, 24, 46, 48] and speech recognition [11, 25]. In particular, in recent works, Reinforcement Learning was utilized in dynamic multi-user MEC systems [9, 36] to cope empirically with the large-scale problems. For instance, in [27, 32], the allocation of the resources and computation task offloading have been defined as a non-linear problem to reduce the entire system cost. Afterwards, Q-learning and Deep-Q-Network-based algorithms are proposed to find the optimal policy that achieves this goal. Similarly, the authors in [41] have developed a reinforcement learning-based algorithm and computation offloading with energy harvesting method for IoT devices with multi-servers which can obtain the optimum offloading strategy based on the status of battery, the capacity of channel, and the amount of energy harvested that will be predicted in the future. In addition, a computation offloading of deep-Q-learning-based approach is designed to decrease the dimension of the state space and speed the learning process.

Furthermore, Sadeghi et al. [45] have proposed a novel framework which finds the optimal caching policy using local and global Markov chains and Q-learning algorithm. Firstly, the requested files' popularity over time is modeled at both local and global scale using Markov. Then, Q-learning-based a linear approximation function algorithm is designed to derive the optimum caching strategy in an online way as well as provide with scalability over the large networks.

Recently, an online proactive caching approach is proposed in [6] where the future requests of the contents will be predicted using deep learning. This approach is composed of three layers in which convolutional neural network is used in the first layer to reduce the complexity of the input data. Then, the bidirectional recurrent neural network is adopted as a second layer to predict the content's requests from mobile device users over time steps. Finally, a fully connected neural network is utilized to improve the prediction performance of the previous layer.

It is seen from the above review of previous studies, computation offloading has been investigated for multi-user MEC system with different objectives and used different solver methods. However, most of these studies are mainly concerns on the content caching while few studies deal with task caching in which considering the computation offloading and task caching has a great effect on reducing the mobile devices' overhead. Moreover, the accomplishment of the optimum strategy is not considered in time-variant systems. This motivates the work of this paper for addressing those considerations in which the task caching with computation offloading is introduced. In addition, Reinforcement Learning algorithms are utilized to derive the best policy and improve the learning speed.

Finally, reducing the energy exhaustion and mobile device's communication delay is the main goal of our work.

3 System model

As outlined in the preceding section, our work considers MEC system with a a multi-user and multi-task, in which $\mathbb{A} = \{1, 2, \ldots, N\}$ represents the mobile users' set, as shown in Fig. 1. Each device has $\mathbb{B} = \{1, 2, \ldots, M\}$ as a set of tasks to be accomplished. In addition, these devices are attached to a single base station that provides the computational and caching services via the MEC server.

The following subsections provide a comprehensive explanation for the communication and computation models which is followed by the task caching model. Afterwards, the formulation for our optimization problem is introduced.

3.1 Communication model

The model of communication is firstly introduced, in which the environment has N users and M computation tasks associated with each device to be completed. In addition, mobile users are connected to a single base station through a wireless channel. Let $x_{i,j} \in \{0, 1\}$ denote the offloading decision. Specifically, if $(x_{i,j} = 0)$, then the tasks of mobile user i can be processed locally. Otherwise, $(x_{i,j} = 1)$, the tasks of mobile user i can be processed remotely at the edge server. So, $X = \{x_{1,1}, x_{1,2}, \ldots, x_{N,M}\}$ is represent the profile of the offloading decision for mobile users.

In this study, Orthogonal Frequency Division Multiple Access method is utilized for addressing simultaneously transmissions of multi-user via the same cell [17]. Thus, each user i can be allocated an uplink rate regrading the following equation [43]:

$$r_i = B \; log_2\left(1 + \frac{p_i g_0^2}{\omega_0 B}\right) \tag{1}$$

where p_i and B are denote the transmission power of mobile i and the system bandwidth, while ω_0 and g_0 are denote the noise power and the channel gain, respectively.

3.2 Computation model

This section presents the model of computation in which the computation tasks for each user can be executed whatever, locally or remotely. Moreover, the computation task's requirement at each user i can be represented using a tuple $\{b_{i,j}, c_{i,j}, \tau_{i,j}\}$, in which $b_{i,j}$, $c_{i,j}$, and $\tau_{i,j}$ are referring to the task offload data, CPU cycles and the completion time deadline which are needed to complete the task j of user i, respectively. The information of $b_{i,j}$ and $c_{i,j}$ can be obtained by using program profilers [38, 58]. These computation tasks may be processed locally or offloaded and processed at the edge server. Hence, the energy and time consumption for running these tasks, whatever locally or remotely, will be explained next in details.

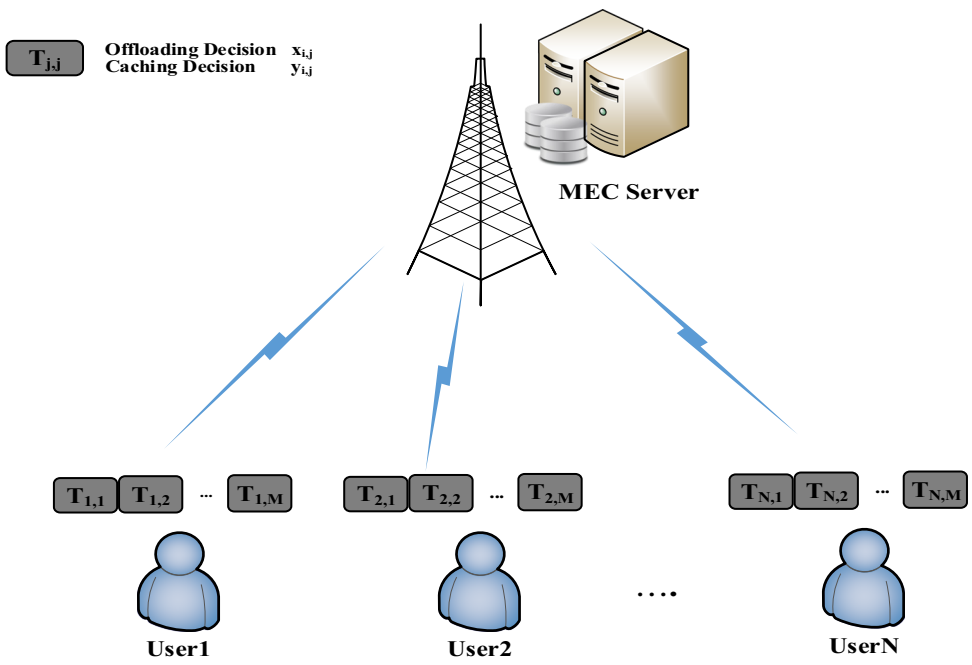

Fig. 1 System model

3.2.1 Energy and time for local execution

In local execution process, the energy and time consumption for completing the task j on the user i can be defined as follows:

$$E_{i,j}^{l} = v_i c_{i,j} \tag{2}$$

$$T_{i,j}^{l} = \frac{c_{i,j}}{f_i^{l}} \tag{3}$$

where f_i^l represents the computation capability of mobile user i (CPU/cycles per second) and v_i is a coefficient, that represents the consumption of energy for each cycle of CPU. In this study, guided by the measurement that obtained in [14, 33, 53], $v_i = 10^{-11}(f_i^l)^2$ is set.

3.2.2 Energy and time for remote execution

In edge server execution process, the mobile user incurs an extra time and energy overhead which is composed of transmitting (i.e., task transmission), receiving the results (i.e., task output) and processing (i.e., task execution) the tasks at the edge server.

In this study, guided by the intuition in [15, 18], the time and energy overhead for receiving the results back from MEC server is neglected where the result's size is ordinarily much smaller than transmitting data. Thus, with regard to Eq. (1), the total cost for completing the task j of the mobile user i remotely can be respectively formulated as follow:

$$T_{i,j}^{e} = T_{i,j}^{off} + T_{i,j}^{exec} = \frac{b_{i,j}}{r_i} + \frac{c_{i,j}}{f_i^{e}} \tag{4}$$

$$E_{i,j}^{e} = E_{i,j}^{off} = p_i T_{i,j}^{off} \tag{5}$$

where f_i^e represents the edge server computation capability which is assigned to mobile user i. Note that, the edge server's computation capability is proportionally assigned to the mobile device according to the mobile device computation capability.

3.3 Task caching

For the task caching model, MEC server is responsible for caching the application program and related code for the completed tasks. However, the MEC server has a finite storage and computing capacity, so the computation capacity and the number of requests[1] for each task are used as a caching strategy to decide which task should be cached. Moreover, the computing and storage capacity of MEC server are denoted by F_s and F_c, respectively.

[1] The requests' number of computation task j is assumed to be known [26].

Furthermore, the process for caching computation tasks is as follows. Firstly, from MEC server perspective, all the information of computation tasks, their related code and the task requests' number are attached. Then, the edge server will find the optimal strategy (i.e., deciding which task will be cached) for caching these tasks while minimizing the time and energy cost of mobile users. Secondly, from the mobile device perspective, each mobile user requests the task to be offloaded and accomplished at edge server. If this task is not cached, then the application program and related code will be offloaded. Otherwise, the computation task will be executed and the output will be returned back. By this way, the cached task does not needed to be offloaded. Thus, caching policy can substantially decrease the time and energy overhead for the mobile user where it is simplified into only the task execution time on the edge server $T_{i,j}^{exec}$.

Let $y_{i,j} \in \{0, 1\}$ represents the caching decision for task j of mobile user i, in which $(y_{i,j} = 0)$ refers that the task j of user i is not cached, and $(y_{i,j} = 1)$ refers that the computation task j of the mobile user i is cached. So, $Y = \{y_{1,1}, y_{1,2}, \ldots, y_{N,M}\}$ is represent the profile of task caching decision for mobile users.

3.4 Problem formulation

Regarding the above discussions, where communication, computation, and task caching model are considered, the energy and time overhead for completing the task j can be respectively defined as follows:

$$T_{i,j} = y_{i,j} T_{i,j}^{exec} + (1 - y_{i,j})\left[(1 - x_{i,j})T_{i,j}^{l} + x_{i,j} T_{i,j}^{e}\right] \tag{6}$$

$$E_{i,j} = (1 - y_{i,j})\left[(1 - x_{i,j})E_{i,j}^{l} + x_{i,j} E_{i,j}^{e}\right] \tag{7}$$

Finally, the cost function of the offloading and caching decision can be obtained and presented as:

$$Z_{i,j} = \beta_i^E E_{i,j} + \beta_i^T T_{i,j} \tag{8}$$

where β_i^E and β_i^T parameter denote the weight between energy and time consumption.

Consequently, regarding the above models, the formulation for our problem can be expressed as follows:

$$
\begin{aligned}
\min_{X,Y} \quad & \sum_{i=1}^{N}\sum_{j=1}^{M} Z_{i,j} \\
\text{s.t} \quad & \sum_{i=1}^{N}\sum_{j=1}^{M} x_{i,j} r_i \le R, && C1 \\
& \sum_{i=1}^{N}\sum_{j=1}^{M} x_{i,j} f_i^e \le F_c, && C2 \\
& \sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j} b_{i,j} \le F_s, && C3 \\
& T_{i,j} \le \tau_{i,j}, && C4 \\
& y_{i,j} \le x_{i,j}, && C5 \\
& x_{i,j} \in \{0,1\}, && C6 \\
& y_{i,j} \in \{0,1\}, && C7
\end{aligned}
\tag{9}
$$

where the objective function minimizes the overhead of time and energy. The constraint C1 limits the channel bandwidth capacity in which R denotes the total uplink data rate. The constraints C2 and C3 characterize the computing and storage upper boundary capacity of MEC server, respectively. The constraint C4 deals with the computation task's deadline requirement. The constraint C5 guarantees that the computation task with local execution decision cannot be cached on the server. The final constraints C6 and C7 guarantee that the computation offloading and task caching decision variables are binary.

The optimal solution for the problem (9) is given where the value for offloading decision vector X and task caching vector Y are obtained. However, for the fact that X and Y variables are binary, then, the feasible set and objective function are not convex. Thus, the problem is a NP-hard [21]. Moreover, solving this type of problem for large users is difficult, because of the problem of curse dimensionality. Consequently, instead of using traditional optimization approaches, reinforcement learning-based methods are utilized to derive the near-optimal values for X and Y in an efficient manner.

4 Problem solution using reinforcement learning-based methods

Reinforcement Learning is one of the machine learning area for making the decisions in an optimal way, in which the agent learns how to behave. The processes for agent to learn and behave is shown in Fig. 2. First, at time step t, the state s_t is detected by the agent from the environment, and an action a_t was chosen from the action space A, then maps from the state s_t to s_{t+1} regarding the probability of choosing the action a_t based on the behavior policy $\pi = P(a_t|s_t)$. Furthermore, a reward r_t was obtained and transitions to a new state s_{t+1} regarding the dynamic environment where the function of reward and the probability of state transition are defined as $R(s, a)$ and $P(s_{t+1}|s_t, a_t)$ respectively [47, 60]. Finally, maximizing the expected cumulative rewards is the main goal of RL where the accumulated rewards is defined as $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ with a discount factor $\gamma \in [0, 1]$.

In order to solve our problem, an equivalent form of reinforcement learning is generated in which the state spaces are represented using all the possible solutions and the actions are defined using the movement among the different states. Then, an efficient Q-learning-based approach is proposed to derive the near-optimum decision where the values for computation offloading and tasks caching variables are obtained. Afterwards, a Deep-Q-Network-based method is developed to address the curse of dimensionality challenge in which Q-learning' action-value function is estimated using deep neural network.

4.1 Reinforcement learning key elements

Reinforcement learning method has the main three key elements used to define the system model, namely, state, action and reward. Therefore, to transform the optimization model into an equivalent reinforcement learning form, these key elements can be expressed as follows:

- *System state* State space is a $1 \times 2NM$ vector which consists of two parts, the offloading decision $X = \{x_{1,1}, x_{1,2}, \ldots, x_{N,M}\}$, and task caching decision $Y = \{y_{1,1}, y_{1,2}, \ldots, y_{N,M}\}$, respectively. Therefore, at an arbitrary index t, the system state can be defined as:

$$
\begin{aligned}
s(t) = \{&x_{1,1}(t), x_{1,2}(t), \ldots, x_{N,M}(t), \\
&y_{1,1}(t), y_{1,2}(t), \ldots, y_{N,M}(t)\}
\end{aligned}
\tag{10}
$$

- *System action* The action is used to denote how the agent can move between two different states. In this paper, the system action is defined as an index-selection from the state vector length which can be applied on the current state to move to a specific neighbouring state.

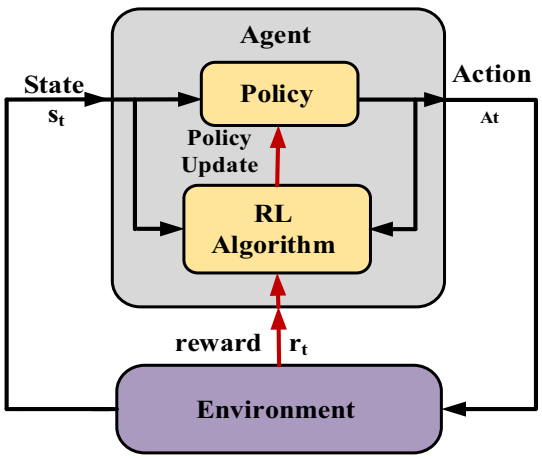

Fig. 2 Reinforcement learning illustration

Specifically, a variable k is define the index selection, in which $k = 1, 2, \ldots, 2NM$, and the action $a(t) = \{a_k(t)\}$ is considered as $1 \times 2NM$ vector.

- *Reward* The reward $R(s, a)$ is a scalar feedback signal that indicates how well the agent is doing in a given state s in accordance with executing the action a at step t. In particular, the objective function of our model and the function of reward should be co-related. In addition, as the value of the objective function is calculated based on $s(t)$. Subsequently, this value can be defined as $V(t)$ which can be defined as follows:

$$V_{s(t)}(t) = (\{x_{i,j}(t)\}, \{y_{i,j}(t)\}) \tag{11}$$

where $\{x_{i,j}(t)\}$ and $\{y_{i,j}(t)\}$ are supposed to be given using the state $s(t)$ that is expressed in Eq. (10). Furthermore, the reward value of state-action pair $(s(t), a(t))$ can be specified on the basis of $V_{s(t)}(t)$ and $V_{s(t+1)}(t+1)$ as follows:

$$r_{s(t),a(t),s(t+1)} = \begin{cases} 1 & V_{s(t)}(t) > V_{s(t+1)}(t+1) \\ -1 & V_{s(t)}(t) < V_{s(t+1)}(t+1) \\ 0 & V_{s(t)}(t) = V_{s(t+1)}(t+1) \end{cases} \tag{12}$$

However, with the growing number of mobile users the state space will rapidly increase. Therefore, a pre-classification iteration is applied to decrease the size of state space before learning, where the computation tasks, that do not satisfy the completion time deadline constraints (i.e., $T_{i,j} > \tau_{i,j}$), should be forced to run locally (i.e, $x_{i,j} = 0$). Besides, the computation tasks with the local decision (i.e., $x_{i,j} = 0$), do not needed to be cached on the edge server (i.e., $y_{i,j} = 0$). Moreover, the possible values of X and Y will be reduced which decrease the state space of the reinforcement learning agent.

4.2 Q-learning algorithm

Q-learning is considered as a model-free algorithm, in which its learning method is based on recording Q value in the form of Q-table [32]. This table is just a simple lookup table of state-action pairs where the system states S are represented by rows and the actions A between them are represented by columns. Therefore, each state-action pair define the quality of an action taken from that state, $Q(s, a)$, regarding a long term reward. Consequently, $Q(s, a)$ value can be computed and updated as follows:

$$Q(s,a) = Q(s,a) + \alpha\Big[R(s,a) + \gamma \max Q(s',a') - Q(s,a)\Big] \tag{13}$$

where $Q(s, a)$ on the left side represents the updated value of that state and action. While, $Q(s, a)$, $R(s, a)$ and $\max Q(s'a')$ represent the current Q value, the received reward value and the maximum expected future reward value, respectively. Further, γ and α represent the discount factor and learning rate, respectively.

In our work, state $s = \{x_{i,j}, y_{y,j}\}$ includes the offloading and caching decisions, respectively. While the corresponding movement among different states is used to define the action a. Consequently, Algorithm 1 shows the proposed Q-learning algorithm process for solving our problem in Eq. (9). In addition, at each step, $Q(s, a)$ value is updated. Consequently, the near-optimum value for offloading and caching decision can be obtained.

Algorithm 1 Q-Learning Algorithm to solve Eq. (9)

```
Initialize Q(s, a).
for each episode do
    Choose a random state s(t).
    Generate a random number rand
    if rand < ϵ then
        select a(t) randomly. {Exploration}
    else
        Set a(t) = max_{a∈A} Q(a). {Exploitation}
    end if
    while s(t) is not terminal do
        Execute chosen a(t), observe reward r(t) according to Eq. (12)
        if V_{s(t)}(t) > V_{s(t+1)}(t + 1) then
            Set r(t) = 1
        else if V_{s(t)}(t) < V_{s(t+1)}(t + 1) then
            Set r(t) = -1
        else
            Set r(t) = 0
        end if
        Q(s(t), a(t)) ← Q(s(t), a(t)) + α[r(t) + γ max_a Q(S(t + 1), a(t)) − Q(s(t), a(t))]
        Let s(t) ← s(t + 1)
    end while
end for
```

4.3 Deep-Q-Network algorithm

In the previous subsection, Q-learning-based approach is used to solve the problem in Eq. (9) by obtaining the largest reward. However, this algorithm is not considered as the ideal algorithm for solving this problem, especially, for the large number of users because the complexity for the problem is rapidly increasing concerning the action and state space size. Subsequently, it becomes difficult for the Q table for computing and storing the corresponding Q value. Hence, solving this problem becomes computational prohibitive with the increasing of the number of state-action pairs.

To address this limitation, Deep-Q-Network is introduced as an effective algorithm where the Q-value function is approximated using neural network. As shown in Fig. 3, firstly, the weight parameters of the evaluation and target Q networks are initialized with random numbers θ and θ'. In addition, the memory D is created with a certain size P. Then, for each episode k, we choose an initial state s^{init}. After that, for each time step t, the evaluation network gets the state $s(t)$ as input and generate the action $a(t)$ on the basis of ϵ-greedy strategy in which a random action $a(t)$ is generated for $\epsilon \in (0, 1)$ probability and an action chosen $a(t) = arg\max_a Q^{pre}(s(t), a(t); \theta)$ for $1 - \epsilon$ probability. Consequently, the reward value $r(t)$ and the next state $s(t + 1)$ can be obtained using Eq. (12) and then replay memory D stores $(s(t), a(t), r(t), s(t + 1))$ as a transition. After that, for the evaluation network updating, a mini-batch random sample of transitions is selected from memory D and the predicted and labeled Q values, Q^{pre} and Q^{lab}, are calculated respectively as $Q(s(t), a(t); \theta)$, $r(t) + \gamma \max'_a Q^{tar}(s(t + 1), a'(t); \theta')$ using evaluation and target networks. Subsequently, mean square error [12] is utilized as loss function to calculate the different loss between these values and gradient decent algorithm [44] is used to minimize this loss. Finally, the parameter value of θ' will be updated every C steps which is utilized to control the update rate. Algorithm 2 shows the process of the proposed Deep Q Network Algorithm in details.

Algorithm 2 Deep Q Network Algorithm to solve Eq. (9)

Initialize the target and evaluation Q network with a random weights θ and θ', respectively. ($\theta' = \theta$)
Initialize replay memory D with capacity P
for each episode $k \leq 1, 2, \ldots, K$ **do**
  Choose an initial state s^{init}
  for each step t **do**
    Generate $rand \in [0, 1]$ randomly.
    if $rand < \epsilon$ **then**
      Select $a(t)$ randomly. {Exploration}
    else
      Set $a(t) = arg\max_a Q^{pre}(s(t), a(t); \theta)$. {Exploitation}
    end if
    Execute the action $a(t)$ and Calculate $V_{s(t)}(t)$ according to Eq. (12)
    if $V_{s(t)}(t) > V_{s(t+1)}(t + 1)$ **then**
      Set $r(t) = 1$
    else if $V_{s(t)}(t) < V_{s(t+1)}(t + 1)$ **then**
      Set $r(t) =$ -1
    else
      Set $r(t) = 0$
    end if
    Save transition $(s(t), a(t), r(t), s(t + 1))$ in memory D
    Extract a sample random mini-batch of transitions from memory D.
    if $s(t + 1)$ is terminal state **then**
      Set $Q^{lab} = r(t)$.
    else
      Calculate the label Q-value Q^{lab}:
$Q^{lab} = r(t) + \gamma \max'_a Q^{tar}(s(t + 1), a'(t); \theta')$.
    end if
    Optimize the parameter θ using gradient descent algorithm which minimize the loss:
$(Q^{lab} - Q^{pre}(s(t), a(t); \theta))^2$.
    Reset $\theta' = \theta$ after each C steps.
  end for
end for

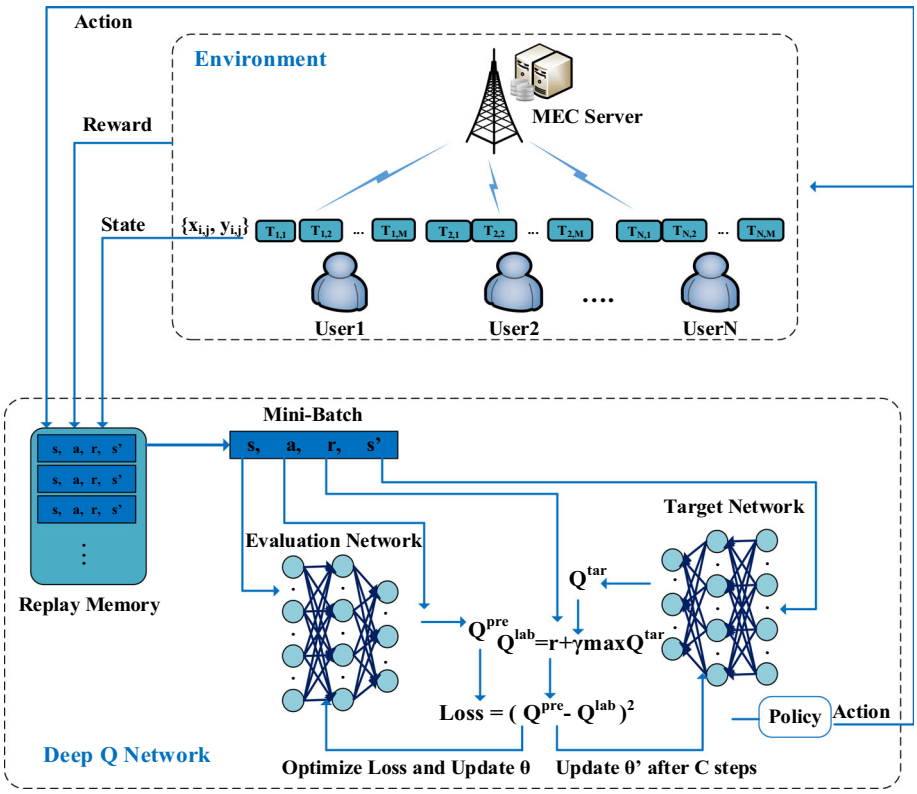

Fig. 3 Deep Q-network (DQN) based MEC system

5 Experiment setup and results

This section undertakes the simulations to validate the design of the proposed model and algorithms. However, we begin by highlighting the environment and resources used in the experiments. Later, we present a discussion on the results reported.

5.1 Experiment setup

In our simulation settings, a real-time video analytic application is used, in which it is considered as one type of killer applications and also requires more computation and processing. As illustrated in Fig. 4, the video analytics application consists of 3 main tasks which are collecting and pre-processing data (collects the cameras' videos and then removes the variations and noise as well as cuts out the normal footage that does not need for further processing), analyzing and tracking data (analyzes and tracks the videos using AI techniques and models), identifying and generating report (identify the objects and generates the final report) [8].

We consider a MEC system with 5 camera devices distributed at different angles and locations connecting to a single base station via a wireless channel which has storage and computing capability associated. Each camera device needs to offload video streaming via a wireless channel to be analyzed and generate a report. A snapshot of the system running is assumed where the offloaded data size and energy consumption per CPU cyle for each mobile user are distributed uniformly within the range of (0, 30) MB and $(0, 20 \times 10^{11})$ J/cycle, respectively. The number of CPU cycles is set 500 cycles/bit. Taking into account the heterogeneous computing capability of camera devices, we randomly assigned each device with $\{0.5, 0.6, \ldots, 1.0\}$ GHz. We also assume that the device' transmission power, the channel bandwidth and Background noise are 0.1 W, 20 MHz and $-$ 100 dBm, respectively. The MEC server' storage and CPU capabilities are set to be 600 MB and 30 GHz, respectively. A simulator in Python is developed using a personal computer with Intel® Core(TM) i7-4770 CPU, 16 GB RAM capacity and 3.4 GHz frequency. Also, Numpy and TensorFlow libraries are utilized to address the Deep Q Network operation where two fully connected layers are used in which 64 units are used for each hidden layer [1]. We set 4000, 2000 and 32 to the number of episode, memory capacity and mini-batch size, whereas, 0.01, 0.99 and 0.1 to learning rate, discount factor and ϵ-greedy value. The other simulation settings employed in the simulations are summarized in Table 1.

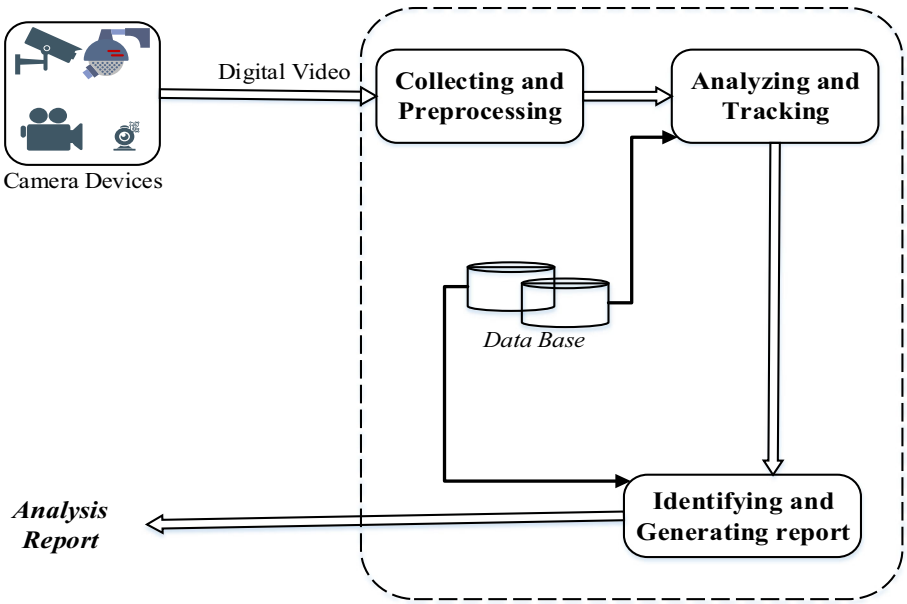

Fig. 4 Video analytic application

Table 1 Simulation parameters

Parameter	Value
Number of devices N	5
Number of tasks M	3
System bandwidth B	20 MHz
Transmission power p_i	100 mW
Background noise ω_0	− 100 dBm
Data size $b_{i,j}$	(0, 30) MB Unif-Dist
Number of CPU cycles per bit $c_{i,j}$	500 Cycles/bit
Computation capability of each device f_i^l	$\{0.5, 0.6, \ldots, 1.0\}$ GHz
Energy consumption per CPU cycle v_i	$(0, 20 \times 10^{-11})$ J/Cycle
Computation capability of edge server F_c	30 GHz
Storage capability of edge server F_s	600 MB
Episode size	4000
Replay memory size	2000
Mini-batch size	32
Learning rate α	0.01
Discount factor γ	0.99
ϵ-greedy value	0.1

5.2 Experiment results

In this section, the convergence performance of our proposed algorithms over different values of learning rate will be studied. Then, the appropriate value will be selected in the next simulation. Afterwards, based on the appropriate values of parameters, our proposed algorithms are compared with different polices to verify their performance.

5.2.1 Convergence performance

Learning rate effect on the convergence performance is demonstrated in Fig. 5, in which the learning rate value is utilized to adapt the updating speed of θ. From Fig. 5, we can see that the convergence process of 0.01 value is faster than 0.001 value. In addition, with increasing the learning rate value the convergence process become faster. However, the convergence process will fall into a local optimal solution for the large value of learning rate (i.e., 0.1). Therefore, an appropriate learning rate should be carefully selected regarding situations. In the next simulations, we set 0.01 as a learning rate value.

5.2.2 Task offloading

This subsection evaluates computation offloading and task caching' effectiveness on the MEC system. Thus, our proposed algorithms are compared with the other two scenarios which are:

- *Local execution* The computation tasks will be processed locally ($x_{i,j} = 0$ and $y_{i,j} = 0$).
- *Full offloading without caching* The computation tasks will be processed remotely ($x_{i,j} = 1$ and $y_{i,j} = 0$).
- *Optimal approach* The combination of task caching and offloading decision (2^{2NM}) are applied and thereafter, the best result is selected. However, this approach can find the global optimal solution but is considered as time-consuming, particularly with increasing the number for mobile users and tasks.

The sum cost of processing the tasks over a different number of devices was presented in Fig. 6(a) and (b),

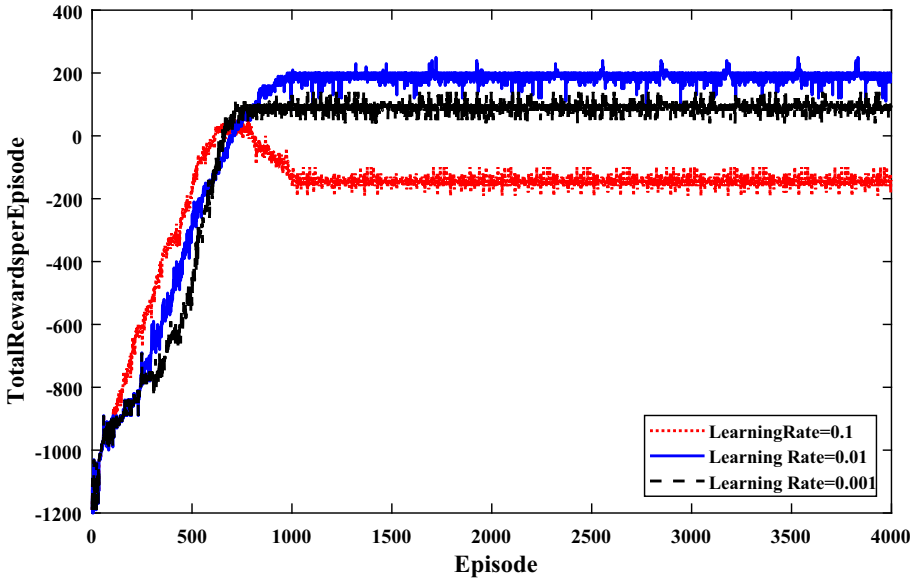

Fig. 5 Convergence performance over different learning rates

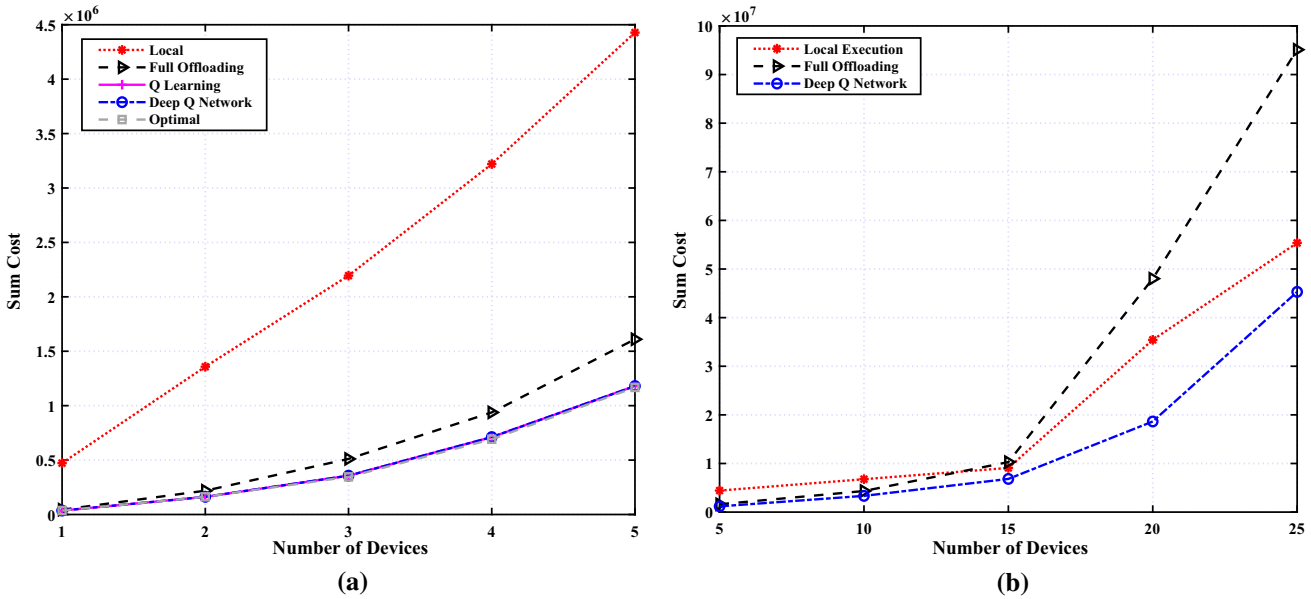

Fig. 6 Sum cost versus number of devices

respectively.[2] It is noticed from Fig. 6 that the sum cost for the proposed algorithms can perform the near-optimal result, i.e., reasonably applying the computation offloading and task caching policy can substantially decrease the total overhead of mobile devices. Specifically, in Fig. 6(a), as the mobile device's number is small (less than 5), the full Offloading scenarios is a little higher than Q learning and Deep Q Network. However, the gap increases rapidly as the number of devices increase which exceeds the local execution values (more than 15) in Fig. 6(b). This is attributed to the edge server computation capability is not quite sufficient for handling many devices simultaneously, so selecting appropriate tasks to be offloaded is considered an important issue under this circumstance.

Figures 7(a) and (b) depict the sum cost of processing the tasks over the average data size, and edge server computation capability, respectively. Firstly, in Fig. 7(a), the sum costs of all scenarios are increasing naturally as the data size of computation tasks increasing, but, the full offloading sum cost value almost becomes equal to the

[2] Deep Q Network is only used in Fig. 6(b) in which the number of devices is increasing and it is becomes difficult with Q table for computing and storing the corresponding Q value as mentioned above.

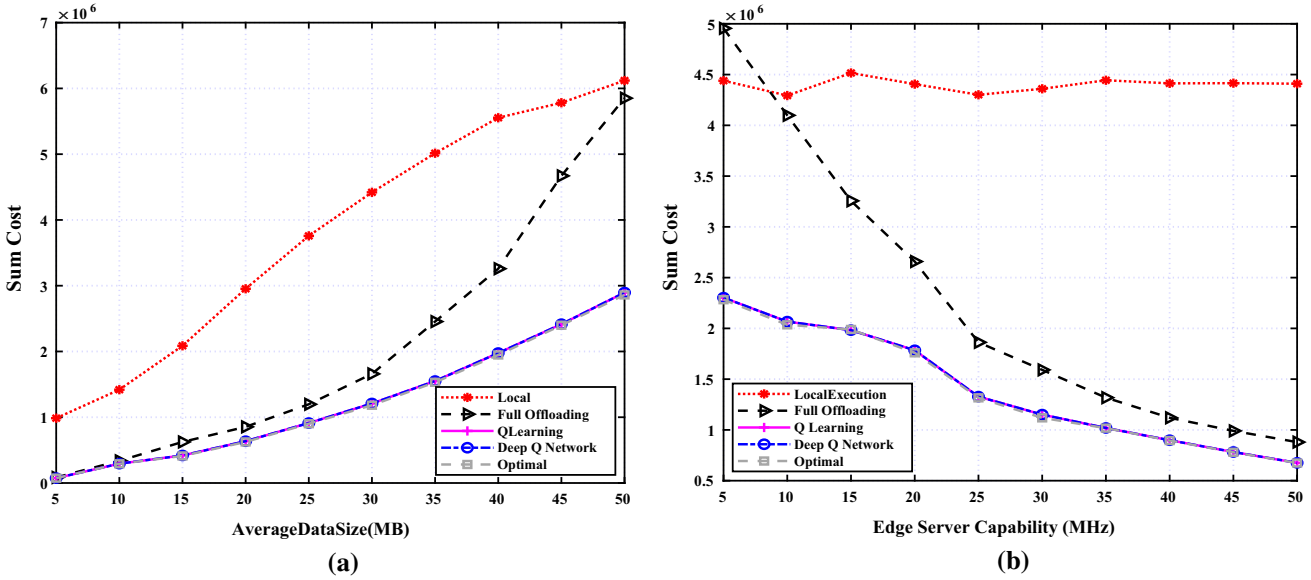

Fig. 7 Sum cost versus data size and edge server capability

local execution policy in the case of 50 MB input data size and this value exceeds the local execution as the data size is becomes greater than 50 MB. This is due to the larger data size leads to consuming more energy and time for the communication process (data transmission) which normally reflected on sum cost. Additionally, in Fig. 7(b), Q learning, Deep Q Network, optimal and full offloading scenarios are decreasing as the edge server computation capability increasing. However, the local execution scenario does not affect. This is due to the processing time becomes shorter as the mobile device can be allocated more resources, while edge server resources are not utilized in the local execution approach.

5.2.3 Task caching

This subsection evaluates the task caching policy for our proposed algorithms in comparison with another scenario which is:

- *Full offloading and random caching (FORC)* All the computation tasks will be executed remotely where some of them are randomly cached at the edge server, till reaching the caching capacity.

Figure 8 illustrates that our proposed algorithms can achieve the best result while FORC scenario the worst one. Specifically, in Fig. 8(a), we can see that increasing the size of caching capacity leads to a decrease in the total overhead, where a larger caching allows for caching more tasks. Besides, Fig. 8(b) proved that our algorithms, Q learning, Deep Q Network, can get the best result. This is due to our proposed algorithms can successfully consider the computation capacity and the number of requests for each task for determining the caching policy while the FORC scenario fails to consider these issues.

6 Conclusion

An efficient computation offloading and task caching model has been proposed in this study for a multi-user with a multi-task MEC system, in which, a new caching concept is applied to the computation tasks where the application program and related code for the completed tasks are cached on the edge server for next execution. In addition, a new model of task offloading and caching integration is formulated as a nonlinear problem whose goal is to reduce the time and energy sum cost. Furthermore, an equivalent form of reinforcement learning is given, to derive the near-optimal solution, where the state space and actions can be defined as the possible solutions and the corresponding movement between different states, respectively. Then, Q-learning and Deep-Q-Network-based algorithms have been proposed which can solve this problem and derive the near-optimum solution in an efficient manner. Finally, the simulation and experimental results proved that the energy and time can be significantly saved when offloading and caching the tasks at the edge server, mainly because of the higher time and energy consumption associated with communication versus computation.

Mobility management for mobile users will be studied in the future, in which the mobile device can be moved dynamically between different base stations through the period of computation offloading. Furthermore, security issues related to the transmission data will be discussed.

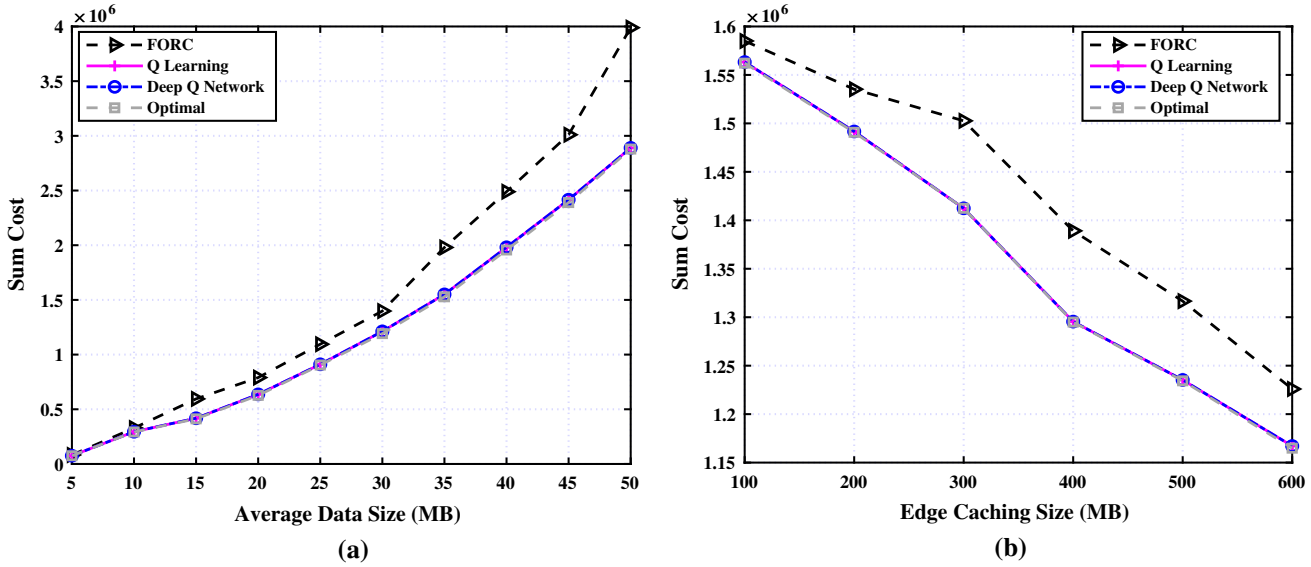

Fig. 8 Sum cost versus data size and edge storage capability

Acknowledgements This work was supported in part by the Key Research and Development Program for Guangdong Province (2019B010136001) and the National Key Research and Development Plan under Grant 2017YFB0801801, in part by the National Natural Science Foundation of China (NSFC) under Grants 61672186 and 61872110, in part by Shenzhen Science and Technology Research and Development Fundation (JCYJ20190806143418198).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. (2016). *Tensorflow: Large-scale machine learning on heterogeneous distributed systems*. arXiv preprint arXiv:1603.04467.
2. Abd El-Latif, A. A., Abd-El-Atty, B., Mazurczyk, W., Fung, C., & Venegas-Andraca, S. E. (2020). Secure data encryption based on quantum walks for 5G internet of things scenario. *IEEE Transactions on Network and Service Management*, *17*(1), 118–131.
3. Abd El-Latif, A. A., Abd-El-Atty, B., Venegas-Andraca, S. E., Elwahsh, H., Piran, M. J., Bashir, A. K., et al. (2020). Providing end-to-end security using quantum walks in IoT networks. *IEEE Access*, *8*, 92687–92696.
4. Abd EL-Latif, A. A., Abd-El-Atty, B., Venegas-Andraca, S. E. and Mazurczyk, W., (2019). Efficient quantum-based security protocols for information sharing and data protection in 5G networks. *Future Generation Computer Systems*, *100*, 893–906.
5. Abou-Nassar, E. M., Iliyasu, A. M., El-Kafrawy, P. M., Song, O.-Y., Bashir, A. K., & Abd El-Latif, A. A. (2020). Ditrust chain: Towards blockchain-based trust models for sustainable healthcare iot systems. *IEEE Access*, *8*, 111223–111238.
6. Ale, L., Zhang, N., Wu, H., Chen, D., & Han, T. (2019). Online proactive caching in mobile edge computing using bidirectional deep recurrent neural network. *IEEE Internet of Things Journal*, *6*, 5520–5530.
7. AlZu'bi, S., Shehab, M., Al-Ayyoub, M., Jararweh, Y., & Gupta, B. (2020). Parallel implementation for 3D medical volume fuzzy segmentation. *Pattern Recognition Letters*, *130*, 312–318.
8. Ananthanarayanan, G., Bahl, V., Cox, L., Crown, A., Nogbahi, S., & Shu, Y. (2019). Video analytics-killer app for edge computing. In *Proceedings of the 17th annual international conference on mobile systems, applications, and services* (pp. 695–696). ACM.
9. Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). *A brief survey of deep reinforcement learning*. arXiv preprint arXiv:1708.05866.
10. Boukerche, A., Guan, S., & Grande, R. E. D. (2019). Sustainable offloading in mobile cloud computing: Algorithmic design and implementation. *ACM Computing Surveys (CSUR)*, *52*(1), 1–37.
11. Chen, J., & Ran, X. (2019). Deep learning with edge computing: A review. *Proceedings of the IEEE*, *107*(8), 1655–1674.
12. Chen, L., Qu, H., Zhao, J., Chen, B., & Principe, J. C. (2016). Efficient and robust deep learning with correntropy-induced loss function. *Neural Computing and Applications*, *27*(4), 1019–1031.
13. Chen, M.-H., Liang, B., & Dong, M. (2017). Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point. In *IEEE INFOCOM 2017-IEEE conference on computer communications* (pp. 1–9). IEEE.
14. Chen, X. (2014). Decentralized computation offloading game for mobile cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, *26*(4), 974–983.
15. Chen, X., Jiao, L., Li, W., & Fu, X. (2015). Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, *24*(5), 2795–2808.
16. Chen, Y., Zhang, N., Zhang, Y., Chen, X., Wu, W., & Shen, X. S. (2019). Toffee: Task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing. *IEEE Transactions on Cloud Computing*, https://doi.org/10.1109/TCC.2019.2923692.
17. Deb, S., & Monogioudis, P. (2015). Learning-based uplink interference management in 4G LTE cellular systems. *IEEE/ACM Transactions on Networking*, *23*(2), 398–411.
18. Elgendy, I. A., Zhang, W., Tian, Y.-C., & Li, K. (2019). Resource allocation and computation offloading with data security for

mobile edge computing. *Future Generation Computer Systems*, *100*, 531–541.
19. Elgendy, I. A., Zhang, W.-Z., Zeng, Y., He, H., Tian, Y.-C., & Yang, Y. (2020). Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile IoT networks. *IEEE Transactions on Network and Service Management*, *17*, 2410–2422.
20. Elgendy, I., Zhang, W., Liu, C., & Hsu, C.-H. (2018). An efficient and secured framework for mobile cloud computing. *IEEE Transactions on Cloud Computing*. https://doi.org/10.1109/TCC.2018.2847347
21. Fooladivanda, D., & Rosenberg, C. (2011). Joint resource allocation and user association for heterogeneous wireless cellular networks. *IEEE Transactions on Wireless Communications*, *12*(1), 248–257.
22. Gad, R., Talha, M., Abd El-Latif, A. A., Zorkany, M., Ayman, E.-S., Nawal, E.-F., et al. (2018). Iris recognition using multi-algorithmic approaches for cognitive internet of things (CIoT) framework. *Future Generation Computer Systems*, *89*, 178–191.
23. Guo, S., Chen, M., Liu, K., Liao, X., & Xiao, B. (2020). Robust computation offloading and resource scheduling in cloudlet-based mobile cloud computing. *IEEE Transactions on Mobile Computing*. https://doi.org/10.1109/TMC.2020.2973993
24. Gupta, B., & Quamara, M. (2020). An overview of internet of things (IoT): Architectural aspects, challenges, and protocols. *Concurrency and Computation: Practice and Experience*, *32*(21), e4946.
25. Han, Y., Wang, X., Leung, V., Niyato, D., Yan, X., & Chen, X. (2019). *Convergence of edge computing and deep learning: A comprehensive survey*. arXiv preprint arXiv:1907.08349.
26. Hao, Y., Chen, M., Hu, L., Hossain, M. S., & Ghoneim, A. (2018). Energy efficient task caching and offloading for mobile edge computing. *IEEE Access*, *6*, 11365–11373.
27. Huang, L., Feng, X., Zhang, C., Qian, L., & Wu, Y. (2019). Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing. *Digital Communications and Networks*, *5*(1), 10–17.
28. Jararweh, Y., Doulat, A., Alqudah, O., Ahmed, E., Al-Ayyoub, M., & Benkhelifa, E. (2016). The future of mobile cloud computing: Integrating cloudlets and mobile edge computing. In *International conference on telecommunications*.
29. Khayyat, M., Alshahrani, A., Alharbi, S., Elgendy, I. A., Paramonov, A., & Koucheryavy, A. (2020). Multilevel service-provisioning-based autonomous vehicle applications. *Sustainability*, *12*(6), 2497–2513.
30. Khayyat, M., Elgendy, I. A., Muthanna, A., Alshahrani, A. S., Alharbi, S., & Koucheryavy, A. (2020). Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks. *IEEE Access*, *8*, 137052–137062.
31. Li, D., Deng, L., Gupta, B. B., Wang, H., & Choi, C. (2019). A novel cnn based security guaranteed image watermarking generation scenario for smart city applications. *Information Sciences*, *479*, 432–447.
32. Li, J., Gao, H., Lv, T., & Lu, Y. (2018). Deep reinforcement learning based computation offloading and resource allocation for MEC. In *2018 IEEE Wireless communications and networking conference (WCNC)* (pp. 1–6). IEEE.
33. Lin, X., Wang, Y., Xie, Q., & Pedram, M. (2015). Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment. *IEEE Transactions on Services Computing*, *8*(2), 175–186.
34. Liu, P., Xu, G., Yang, K., Wang, K., & Meng, X. (2018). Jointly optimized energy-minimal resource allocation in cache-enhanced mobile edge computing systems. *IEEE Access*, *7*, 3336–3347.
35. Liu, Y., Peng, J., Kang, J., Iliyasu, A. M., Niyato, D., & El-Latif, A. A. A. (2020). A secure federated learning framework for 5G networks. arXiv preprint arXiv:2005.05752.
36. Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y.-C., et al. (2019). Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys & Tutorials*, *21*, 3133–3174.
37. Lyu, X., Ren, C., Ni, W., Tian, H., Liu, R. P., & Tao, X. (2020). Distributed online learning of cooperative caching in edge cloud. *IEEE Transactions on Mobile Computing*. https://doi.org/10.1109/TMC.2020.2983924
38. Lyu, X., & Tian, H. (2016). Adaptive receding horizon offloading strategy under dynamic environment. *IEEE Communications Letters*, *20*(5), 878–881.
39. Mach, P., & Becvar, Z. (2017). Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys and Tutorials*, *19*(3), 1628–1656.
40. Mao, Y., Zhang, J., Song, S., & Letaief, K. B. (2016). Power-delay tradeoff in multi-user mobile-edge computing systems. In *2016 IEEE Global communications conference (GLOBECOM)* (pp. 1–6). IEEE.
41. Min, M., Xiao, L., Chen, Y., Cheng, P., Wu, D., & Zhuang, W. (2019). Learning-based computation offloading for IoT devices with energy harvesting. *IEEE Transactions on Vehicular Technology*, *68*(2), 1930–1941.
42. Nur, F. N., Islam, S., Moon, N. N., Karim, A., Azam, S., & Shanmugam, B. (2019). Priority-based offloading and caching in mobile edge cloud. *Journal of Communications Software and Systems*, *15*(2), 193–201.
43. Rappaport, T. S. (2009). *Wireless communications: Principles and practice*. London: Prentice Hall.
44. Ruder, S. (2016). *An overview of gradient descent optimization algorithms*. arXiv preprint arXiv:1609.04747.
45. Sadeghi, A., Sheikholeslami, F., & Giannakis, G. B. (2017). Optimal and scalable caching for 5G using reinforcement learning of space-time popularities. *IEEE Journal of Selected Topics in Signal Processing*, *12*(1), 180–190.
46. Stergiou, C. L., Psannis, K. E., & Gupta, B. B. (2020). IoT-based big data secure management in the fog over a 6G wireless network. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2020.3033131
47. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. Cambridge: MIT Press.
48. Tewari, A., & Gupta, B. (2020). Security, privacy and trust of different layers in internet-of-things (IoTs) framework. *Future Generation Computer Systems*, *108*, 909–920.
49. Vallina-Rodriguez, N., & Crowcroft, J. (2013). Energy management techniques in modern mobile handsets. *IEEE Communications Surveys Tutorials*, *15*(1), 179–198.
50. Wang, H., Li, R., Fan, L., & Zhang, H. (2017). Joint computation offloading and data caching with delay optimization in mobile-edge computing systems. In *International conference on wireless communications and signal processing* (pp. 1–6).
51. Wang, H., Li, Z., Li, Y., Gupta, B., & Choi, C. (2020). Visual saliency guided complex image retrieval. *Pattern Recognition Letters*, *130*, 64–72.
52. Wang, S., Zhang, X., Zhang, Y., Wang, L., Yang, J., & Wang, W. (2017). A survey on mobile edge networks: Convergence of computing, caching and communications. *IEEE Access*, *5*(99), 6757–6779.
53. Wen, Y., Zhang, W., & Luo, H. (2012). Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones. In *IEEE INFOCOM* (pp. 2716–2720).
54. Wang, X., Han, Y., Leung, V. C., Niyato, D., Yan, X., & Chen, X. (2020). Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, *22*, 869–904.
55. Yadav, R., Zhang, W., Kaiwartya, O., Singh, P. R., Elgendy, I. A., & Tian, Y.-C. (2018). Adaptive energy-aware algorithms for

minimizing energy consumption and sla violation in cloud computing. *IEEE Access*, *6*, 55923–55936.
56. Yang, P., Zhang, N., Zhang, S., Yu, L., Zhang, J., & Shen, X. (2019). Content popularity prediction towards location-aware mobile edge caching. *IEEE Transactions on Multimedia*, *21*(4), 915–929.
57. Yi, C., Cai, J., & Su, Z. (2020). A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications. *IEEE Transactions on Mobile Computing*, *19*(1), 29–43.
58. Zhang, W., Wen, Y., & Wu, D. O. (2014). Collaborative task execution in mobile cloud computing under a stochastic wireless channel. *IEEE Transactions on Wireless Communications*, *14*(1), 81–93.
59. Zhang, W.-Z., Elgendy, I. A., Hammad, M., Iliyasu, A. M., Du, X., Guizani, M., et al. (2020). Secure and optimized load balancing for multi-tier IoT and edge-cloud computing systems. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2020.3042433
60. Zhu, H., Cao, Y., Wang, W., Jiang, T., & Jin, S. (2018). Deep reinforcement learning for mobile edge caching: Review, new features, and open issues. *IEEE Network*, *32*(6), 50–57.

Ibrahim A. Elgendy received his M.Sc. degree from Computer Science Department, Faculty of Computers and Information, Menoufia University, Egypt, in 2016. He worked as assistant lecturer in Faculty of Computers and Information, Menoufia University, Egypt since December 2011 till now. He is currently pursuing the Ph.D. degree from the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China. His research interests are primarily in cloud computing, mobile edge computing and distributed computing.

Wei-Zhe Zhang is currently a professor in the School of Computer Science and Technology at Harbin Institute of Technology, China, and director in the Cyberspace Security Research Center, Pengcheng Laboratory, Shenzhen, China. His research interests are primarily in cyberspace security, cloud computing, and high-performance computing. He has published more than 130 academic papers in journals, books, and conference proceedings. He is a senior member of the IEEE and a lifetime member of the ACM.

Hui He received the Ph.D. degree from the Department of Computer Science, Harbin Institute of Technology, China. She is currently a Full Professor with school of Computer Science and Technology, Harbin Institute of Technology, China. Her research interests are mainly focused on distributed computing, IoT, and big data analysis.

Brij B. Gupta received Ph.D. degree from Indian Institute of Technology Roorkee, India in the area of Information and Cyber Security. In 2009, he was selected for Canadian Commonwealth Scholarship awarded by Government of Canada. He published more than 250 research papers in International Journals and Conferences of high repute. His biography was selected and published in the 30th Edition of Marquis Who's Who in the World, 2012. Dr. Gupta also received Young Faculty research fellowship award from Ministry of Electronics and Information Technology, government of India in 2017. He is also working as principal investigator of various R&D projects. He served as associate editor of IEEE Access, IEEE TII, IJICS, etc. He is also serving as reviewer for Journals of IEEE, Springer, Wiley, Taylor & Francis, etc. He was also visiting researcher with Yamaguchi University, Japan, with Deakin University, Australia and with Swinburne University of Technology, Australia during January, 2015 and 2018, July 2017, and March–April 2018, respectively. Moreover, he was also visiting Professor in University of Murcia, Spain in June–July, 2018. Additionally, he was visiting professor with Temple university, USA and Staffordshire University, UK during June, 2019 and July 2020, respectively. At present, Dr. Gupta is working as Assistant Professor in the Department of Computer Engineering, National Institute of Technology Kurukshetra India. His research interest includes Information security, Cyber Security, Cloud Computing, Web security, Intrusion detection and Phishing.

Ahmed A. Abd El-Latif received the B.Sc. degree with honor rank in Mathematics and Computer Science in 2005 and M.Sc. degree in Computer Science in 2010, all from Menoufia University, Egypt. He received his Ph.D. degree in Computer Science and Technology at Harbin Institute of Technology (H.I.T), Harbin, P. R. China in 2013. He is an associate professor of Computer Science at Menoufia University, Egypt and School of Information Technology and Computer Science, Nile University, Egypt. He is author and co-author of more than 140 papers, including refereed IEEE/ACM/Springer/Elsevier journals, conference papers, and book chapters. He received many awards, State Encouragement Award in Engineering Sciences 2016, Arab Republic of Egypt; the best Ph.D. student award from Harbin Institute of Technology, China 2013; Young scientific award, Menoufia University, Egypt 2014. He is a fellow at Academy of Scientific Research and Technology, Egypt. His areas of interests are multimedia content encryption, secure wireless communication, IoT, applied cryptanalysis, perceptual cryptography, secret media sharing, information hiding, biometrics, forensic analysis in digital images, and quantum information processing. Dr. Abd El-Latif has many collaborative scientific activities with international teams in different research projects. Furthermore, he has been reviewing papers for 120+ International Journals including IEEE Communications Magazine, IEEE Internet of Things journal, Information Sciences, IEEE Transactions on Network and Service Management, IEEE Transactions on Services Computing, Scientific reports Nature, Journal of Network and Computer Applications, Signal processing, Cryptologia, Journal of Network and Systems Management, Visual Communication and Image Representation , Neurocomputing, Future Generation Computer Systems, etc. Dr. Abd El-Latif is an associate editor of Journal of Cyber Security and Mobility, and IET Quantum Communication. Dr. Abd El-Latif also led many special issues in several SCI/EI journals.