Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Task offloading optimization mechanism based on deep neural network in edge-cloud environment

Lingkang Meng[1], Yingjie Wang[1*], Haipeng Wang[2], Xiangrong Tong[1], Zice Sun[1] and Zhipeng Cai[3]

Abstract

With the rise of edge computing technology and the development of intelligent mobile devices, task offloading in the edge-cloud environment has become a research hotspot. Task offloading is also a key research issue in Mobile CrowdSourcing (MCS), where crowd workers collect sensed data through smart devices they carry and offload to edge-cloud servers or perform computing tasks locally. Current researches mainly focus on reducing resource consumption in edge-cloud servers, but fails to consider the conflict between resource consumption and service quality. Therefore, this paper considers the learning generation offloading strategy among multiple Deep Neural Network(DNN), proposed a Deep Neural Network-based Task Offloading Optimization (DTOO) algorithm to obtain an approximate optimal task offloading strategy in the edge-cloud servers to solve the conflict between resource consumption and service quality. In addition, a stack-based offloading strategy is researched. The resource sorting method allocates computing resources reasonably, thereby reducing the probability of task failure. Compared with the existing algorithms, the DTOO algorithm could balance the conflict between resource consumption and service quality in traditional edge-cloud applications on the premise of ensuring a higher task completion rate.

Keywords Edge-cloud, Task offloading, Mobile crowdsourcing, Deep neural network, Resource consumption, Service quality

Introduction

With the rapid development of Internet of Things (IoT) [1] and 5G technology [2], edge-cloud has been integrated into daily applications, MCS, as a new mode of perception network, data collection [3] and information service, has become an indispensable part of today's society [4]. MCS is a process in which crowd workers form an interactive perception network by carrying mobile devices to a designated location for information collection and crowdsourcing platforms. MCS is a process in which crowd workers form an interactive perception network by carrying mobile devices to a designated location for information collection and crowdsourcing platforms. The crowdsourcing platform publishes tasks and recruits crowd workers to complete the tasks [5], which provides many conveniences to people's lives, such as collecting information, analyzing data [6], and sharing knowledge [7], so it has received extensive attention in various fields. Academics at Zurich University [8] designed an environmental monitoring model that uses a smartphone carrying a sensor to detect ozone levels. The Smart City project in Serbia [9] uses sensors provided by Libelium [10] on public transport equipment to monitor air quality. In addition, the famous Waze company also provides commercial map services for people based on the MCS model. Mobile crowdsourcing has become a research hotspot in Infocom, Ubicomp, Percom, and Mobicom [11].

*Correspondence:
Yingjie Wang
towangyingjie@163.com
[1] School of Computer and Control Engineering, Yantai University, Yantai, China
[2] Institute of Information Fusion, Naval Aviation University, Yantai, China
[3] Department of Computer Science, Georgia State University, Atlanta, USA

Springer Open

© The Author(s) 2023. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

In a typical mobile crowdsourcing system, a complete cloud-based architecture consists of the cloud platform, task requesters, and crowd workers. First, the task requester issues the task through the cloud platform. Then, select appropriate workers to assign tasks through task assignment [12]. Crowd workers collect data through their mobile devices and upload it to the cloud platform. Finally, the platform will evaluate and update the worker's reputation value [13–15] based on the quality of the worker's uploaded data [16]. With the emergence of technologies such as intelligent driving [17], task requesters have higher and higher requirements for real-time data [18, 19], and workers uploading data to the cloud platform will generate large data delays [20], and traditional centralized cloud platforms will not be able to meet this requirement. The emergence of edge-cloud technology has temporarily solved this problem.

Edge-Cloud refers to the use of an edge-cloud server that integrates network, computing, storage, and application core capabilities on the side close to the mobile device or data source [21] to provide the nearest computing service nearby. When workers use mobile devices to upload data, they can directly interact with the nearest edge node, which greatly reduces data transmission latency [22]. In the edge-cloud environment, computing tasks are performed on a powerful edge-cloud server, which has the advantages of easy installation and small size [23]. But their load capacity and computing power are still far inferior to cloud servers. Chen et al. [24] proposed a game theory-based task offloading algorithm, but this algorithm requires multiple interactions between crowd workers and edge servers, which consumes a lot of resources. Huang et al. [25] proposed a task offloading and resource allocation scheme based on a deep Q-network, but its feature of searching in tables is not suitable for processing high-dimensional data. Therefore, the problem of optimizing the task offloading strategy in edge servers needs to be solved urgently [26]. The main challenges for task offloading in MCS are as follows.

1. In the practical application of MCS, workers often choose the nearest edge-cloud server to upload sensory data. If there are a large number of workers near the edge-cloud server and most workers choose to offload tasks to the edge-cloud server, the edge-cloud server may suffer from excessive data processing capacity. Large and overloaded and leads to paralysis. Therefore, how to make reasonable task offloading decisions is an important research content to prevent excessive load on edge-cloud servers.
2. Although researchers have proposed many schemes to solve the task distribution problem among multiple edge-cloud servers, the computing power of edge-cloud servers is limited. If there are too many tasks in the task queue, some time-sensitive tasks may not be solved in time. Therefore, how to reasonably allocate the computing resources on the edge-cloud server is a key factor to improve the success rate of task allocation.

In response to the above challenges, this paper studies a task offloading optimization algorithm DTOO for the MCS, which generates a near-optimal task offloading strategy and solves the conflict between resource consumption and quality of service. A resource allocation scheme is designed to improve the success rate of the task. The main contributions of this paper are summarized as follows.

1. This paper designs a task offloading algorithm DTOO based on DNN, which can obtain an approximate excellent offloading strategy through learning among multiple neural units, so as to solve the conflict between edge-cloud server resource consumption and service quality.
2. This paper proposes a stack-based resource sorting scheme, which matches different computing resources according to the timeliness level of tasks, thereby improving the success rate of tasks.
3. The proposed DNN-based task offloading scheme and stack-based task ranking mechanism are analyzed and evaluated through comparison experiments on real datasets. The experimental results verified the superiority of this scheme.

The rest of this paper is organized as follows. Section II introduces the related works. Section III describes the DTOO algorithm and resource allocation scheme. Section IV presents the comparison experiments and the discussion of the experimental results. Finally, Section V presents the conclusion.

Related work

In recent years, more and more attention has been paid to the research of task assignment [27, 28] based on mobile crowdsourcing in the edge cloud environment, aiming at design an optimal task offloading strategy with low latency, low energy consumption, and high service quality. Many scholars have conducted in-depth research on this and proposed feasibility studies.

Edge computing

With the popularization of the IoT and the promotion of cloud services [29], edge computing has emerged as a new computing paradigm. Edge computing refers to delegating data processing to the edge of the network

[30]. This mode can reduce request delay and network bandwidth while ensuring data security and privacy [31–33]. The core part of edge computing is to migrate some or all of the computing tasks in cloud computing to the vicinity of mobile devices. This highly potential way can solve some of the shortcomings of cloud computing [34]. Zhao et al. [35] designed a new mobile device data transmission scheme, introduced edge computing in the cloud platform-centric architecture, and used edge nodes to assist data transmission to solve the problem of excessive bandwidth consumption in traditional cloud platform solutions [36]. This scheme explores the bandwidth consumption of edge nodes through the edge computing paradigm. Ren et al. [37] explored the problem of joint communication technology and computing resource allocation, which aimed at find an optimal solution to minimize latency in the cloud and edge-cloud collaborative systems. And esigned an offloading scheme based on distributed computing, which can achieve excellent computing offloading ability and can make corresponding adjustments with the change of user scale [38]. Optimized the problem of multi-user resource offloading of edge cloud in a multi-channel wireless interference environment.

Task offloading based on edge cloud

The problem of offloading computing tasks in edge computing is a research hotspot [39]. In the actual crowdsourcing environment, task offloading will be affected by various external factors, such as the hardware performance of the device, the network environment where the worker is located [40, 41], and the worker's personalized choice [42]. This makes it particularly important to formulate a reasonable and dynamically changing task offloading strategy according to the external environment. Some existing works mainly study how to make task assignment decisions in an offline or online state, and most of the research focuses on minimizing task completion time and resource consumption as the optimization goal. For example, Dinh et al. [43] considered two cases of whether the CPU frequency of the edge server can be adjusted or not, and proposed a linear relaxation-based method and an exhaustive search-based scheme to solve the two cases, respectively. Obviously, the exhaustive approach consumes a lot of computing resources. To get a balance between resource consumption and computational latency. Wu et al. [44] proposed a task offloading algorithm based on Lyapunov, which reduces the resource consumption of the device under the condition of satisfying the delay constraint. Considering service heterogeneity, unknown system dynamics, spatial demand coupling, and decentralized coordination, Xu et al. [45] proposed an online task offloading algorithm based on Lyapunov optimization and Gibbs sampling. Shu et al. [46] designed an algorithm that supports multi-user task offloading, dividing tasks into subtasks and offloading them to edge servers to reduce the end-to-end task execution time. Mao et al. [47] studied an Energy Harvesting (EH) technology to power mobile devices through renewable energy. Based on Lyapunov, the frequency and transmit power of the CPU are optimized to reduce the execution delay of the task. Zhao et al. [48] optimized the offloading decision, radio resource allocation, and computing resource allocation, and transformed the resource minimization problem into the Mixed Integer Nonlinear Programming (MINLP) problem. A Gini coefficient-based greedy heuristic (GCGH) was proposed to solve this problem. Although computing offloading in edge computing is the core technology, how to allocate resources to improve the task completion rate should also be considered in practical crowdsourcing applications.

Deep learning

As an emerging technology in machine learning algorithms, Deep Learning's main purpose is to build and simulate a neural network for analyzing and learning the human brain [49]. The essence of deep learning is to perform hierarchical feature representation on data, and further abstract low-level features into high-level features through neural networks. DNN composed of multi-layer perceptions have achieved major breakthroughs in the fields of image classification and recognition, natural language processing, and a robot control. Mnih et al. [50] used deep neural networks to develop a novel surrogate model called a deep Q-network [51], which bridges the gap between high-level sensory input and decision-making actions. Deep learning is also widely used in the field of wireless communication, such as resource allocation problems [52], signal detection problems [53], data caching problems [54], etc. In recent years, some scholars have used deep learning models to solve the task offloading problem in the edge cloud environment. Huang et al. [55] proposed a deep reinforcement learning-based method to solve the task offloading and resource allocation problem, with the aim of making each user obtain a satisfactory task offloading decision and resource allocation scheme. However, the search nature of deep Q-learning based on Q-table makes its performance not outstanding when dealing with high-dimensional data.

Existing optimization schemes do not take into account the limitation of computational dimensions,

and general optimization algorithms cannot efficiently deal with the complexity of data in the actual crowdsourcing environment, especially when faced with a large number of crowd workers. The existing work failed to consider the allocation of resources when optimizing the task offloading scheme. Therefore, according to the above problems, this paper considers the balance between resource consumption and quality of service, and the task completion rate. A DTOO algorithm is proposed to specify an efficient offloading decision through mutual learning among DNNs. In order to improve the task completion rate and allocate resources according to the priority of tasks, a stack-based resource allocation scheme is designed.

System design

In this section, first, the task offloading problem in the edge cloud environment is defined, then, the system model of this paper is introduced, and finally, the algorithm proposed in this paper is described in detail. The main symbol definitions are shown in Table 1.

Problem definition

Definition 1 (Crowd Task): In MCS system, the crowd tasks are uploaded to the platform by the task requester, and the crowd tasks are released by the platform, defined as $C = \{c_1, c_2, c_3, \ldots, c_n\}$ .Each task also has its properties, where the task is published is defined as $l = \{l_{c_1}, l_{c_2}, l_{c_3}, \ldots l_{c_n}\}$. The time of task release is defined as T_{satrt} . The task deadline is defined as T_{end} . Therefore, the maximum allowable delay of a task can be defined as $T = \{t_{c_1}, t_{c_2}, t_{c_3}, \ldots, t_{c_n}\}$. The data volume of the task is defined as $D = \{d_{c_1}, d_{c_2}, d_{c_1}, \ldots, d_{c_n}\}$.

Definition 2 (Crowd Work): Crowd workers can collect data with their own mobile devices and upload the data to the crowd platform. Crowd workers are defined as $W = \{w_1, w_2, w_3, \ldots, w_m\}$. Each crowd worker also has its attributes, the *id* of the worker is defined as W_{id} .

Definition 3 (Task Offload Policy): For each crowd worker, he or she can choose to process computing tasks locally or offload computing tasks to edge servers for processing. Therefore, considering the task offloading strategy as a binary problem, when workers choose to process computing tasks locally, it is recorded as 0, and when they choose to offload tasks to edge servers for processing, it is recorded as 1. Therefore, the task offloading strategy is defined as Eq. (1):

$$S_{c_i w_j} = \begin{cases} 0, & local \\ 1, & offload \end{cases} \tag{1}$$

where $S_{c_i w_j}$ represents worker w_j choice of offloading strategy for task c_i. $S_{c_i w_j} = 0$, indicates that the task is executed locally, and $S_{c_i w_j} = 1$, indicates that the task is offloaded to the edge server for execution.

Table 1 Symbols and definitions

Notation	Description
$C = \{c_1, c_2, c_3, \ldots, c_n\}$	Crowd task set
$l = \{l_{c_1}, l_{c_2}, l_{c_3}, \ldots l_{c_n}\}$	Crowd tasks posted location set
T_{satrt}	Crowd task release time
T_{end}	Crowd task deadline
$T = \{t_{c_1}, t_{c_2}, t_{c_3}, \ldots, t_{c_n}\}$	Maximum allowable delay set for crowd tasks
$D = \{d_{c_1}, d_{c_2}, d_{c_1}, \ldots, d_{c_n}\}$	The amount of data contained set in the crowd task
$W = \{w_1, w_2, w_3, \ldots, w_m\}$	Crowd worker set
W_{id}	Crowd worker id
$S_{c_i w_j}$	Task offload policy
r_{local}	Local data processing rate
q_{local}	Consumption per bit of data processed locally
x_{edge}	Edge server data transfer rate
r_{edge}	The rate at which edge server data is processed
y_{edge}	Transmission energy consumption per bit data of edge server
q_{edge}	Energy consumption per bit of data processed by edge servers
$T_{local}(c_i, w_j)$	Local time consumption of worker w_j task c_i
$E_{local}(c_i, w_j)$	Local energy consumption of worker w_j task c_i
$T_{edge}(c_i, w_j)$	The marginal time consumption of worker w_j task c_i
$E_{edge}(c_i, w_j)$	The marginal energy consumption of worker w_j task c_i
W_{local}	The total consumption of local processing computing tasks
W_{edge}	The total consumption of edge servers processing computing tasks

Definition 4 (Local Computation): Model situations where users choose to process computing tasks locally. For computation tasks executed locally, the time consumption is defined as Eq. (2):

$$T_{local}(c_i, w_j) = \frac{d_{c_i}}{r_{local}} \tag{2}$$

where $T_{local}(c_i, w_j)$ is the time consumption of worker w_j task c_i, and r_{local} is the rate at which data is processed locally. Energy consumption for local processing is defined as Eq. (3):

$$E_{local}(c_i, w_j) = d_{c_i} \times q_{local} \tag{3}$$

where $E_{local}(c_i, w_j)$ is the energy consumption of worker $w_j taskc_i$, and $q_l ocal$ is the consumption per bit of data

processed locally. Therefore, the joint offloading strategy defines the total consumption of locally processed computing tasks as Eq. (4):

$$W_{local} = \sum_{i=1}^{n}\sum_{j=1}^{m}\left[T_{local}(c_i, w_j) + E_{local}(c_i, w_j)\right] \times \left(1 - S_{c_i w_j}\right) \quad (4)$$

Definition 5 (Edge Computing): Model the user's choice to offload computing tasks to edge servers. The set of edge servers is defined as $ES = \{es_1, es_2, es_3, \ldots, es_z\}$, and the time consumption for computing tasks offloaded to edge servers is defined as Eq. (5):

$$T_{edge}(c_i, w_j) = \frac{d_{c_i}}{x_{edge}} + \frac{d_{c_i}}{r_{edge}} \quad (5)$$

where $T_{edge}(c_i, w_j)$ is the time consumption of worker w_j task c_i, x_{edge} is the data transmission rate of the edge server, and r_{edge} is the data processing rate of the edge server. The energy consumption of offloading computing tasks to edge servers is defined as Eq. (6):

$$E_{edge}(c_i, w_j) = d_{c_i} \times y_{edge} + d_{c_i} \times q_{edge} \quad (6)$$

where $E_{edge}(c_i, w_j)$ is the energy consumption of worker w_j task c_i, y_{edge} is the energy consumption of data transmission per bit, and q_{edge} is the energy consumption of edge server processing each bit of data. Therefore, the joint offloading strategy defines the total consumption of edge server processing computing tasks as Eq. (7):

$$W_{edge} = \sum_{i=1}^{n}\sum_{j=1}^{m}\left[T_{edge}(c_i, w_j) + E_{edge}(c_i, w_j)\right] \times S_{c_i w_j} \quad (7)$$

Then the total consumption of the system is defined as Eq. (8):

$$\begin{aligned} W_{total} = &\sum_{i=1}^{n}\sum_{j=1}^{m}\left[T_{local}(c_i, w_j) + E_{local}(c_i, w_j)\right] \times \left(1 - S_{c_i w_j}\right) \\ &+ \sum_{i=1}^{n}\sum_{j=1}^{m}\left[T_{edge}(c_i, w_j) + E_{edge}(c_i, w_j)\right] \times S_{c_i w_j} \end{aligned} \quad (8)$$

In short, to minimize the resource consumption of task completion, the goal of this stage is to find an optimal task offloading strategy to minimize W_{total}.

Definition 6 (Resource Allocation): Model resource allocation among edge servers. Due to the limited computing power and load capacity of edge servers, and crowd tasks are also time-sensitive, edge servers need to complete computing tasks within a certain period. Therefore, according to the maximum allowable delay of the tasks, this paper sorts the tasks by priority and stores them in the task stack, which corresponds to the address stack in the edge server. When task c_k joins, first determine whether the total delay $\sum_{u=1}^{k-1} t_{c_u}$ of its previous task will exceed the maximum allowable delay of task c_k, and if it exceeds, it will be allocated to other idle addresses.

System model design

In the edge cloud environment, the computing tasks of the central cloud are sunk to the edge of the network, which greatly reduces network latency. At the same time, this paper uses mutual learning among multiple DNNs to obtain an approximate optimal offloading strategy, which also ensures the service quality under the premise of low resource consumption. A stack-based sorting mechanism to reasonably allocate resources is used to improve task completion rate. In practical crowd applications, each crowd worker will have multiple jobs that need to be processed locally or at the edge server, and the offloading decision is represented by 0 or 1. $S_{c_i w_j} = 0$ means that the task is executed locally. $S_{c_i w_j} = 1$ indicates that the task is offloaded to the edge server for execution. The system model is shown in Fig. 1. First, a DNN is used to generate candidate unloading actions by taking the task scale carried by crowd workers as input to the model. Then, an offloading strategy that meets the optimization objective is selected as the output. For computing tasks that are offloaded to edge servers for processing, the tasks are sorted according to their priority. In the task stack, tasks with larger maximum allowable delays are placed at the bottom of the stack, and tasks with lower delays are placed at the top of the stack. In the address stack, servers with more resources, that is, addresses with higher idle levels, are placed at the top of the stack, and those with fewer resources are placed at the bottom of the stack. The task stack and the address stack are arranged in order so that the most urgent tasks can be allocated addresses of more resources, which greatly improves the success rate of the tasks.

Since there may be crowd workers at the boundary of the edge server service range in the candidate set and not recruited by the platform, these workers may have better utility. Therefore, before recruiting workers, the movement trajectories of the workers are predicted to accurately locate the partitions where the workers are located based on the workers' historical trajectories and social networks. Then, assign all the recruited workers to edge servers.

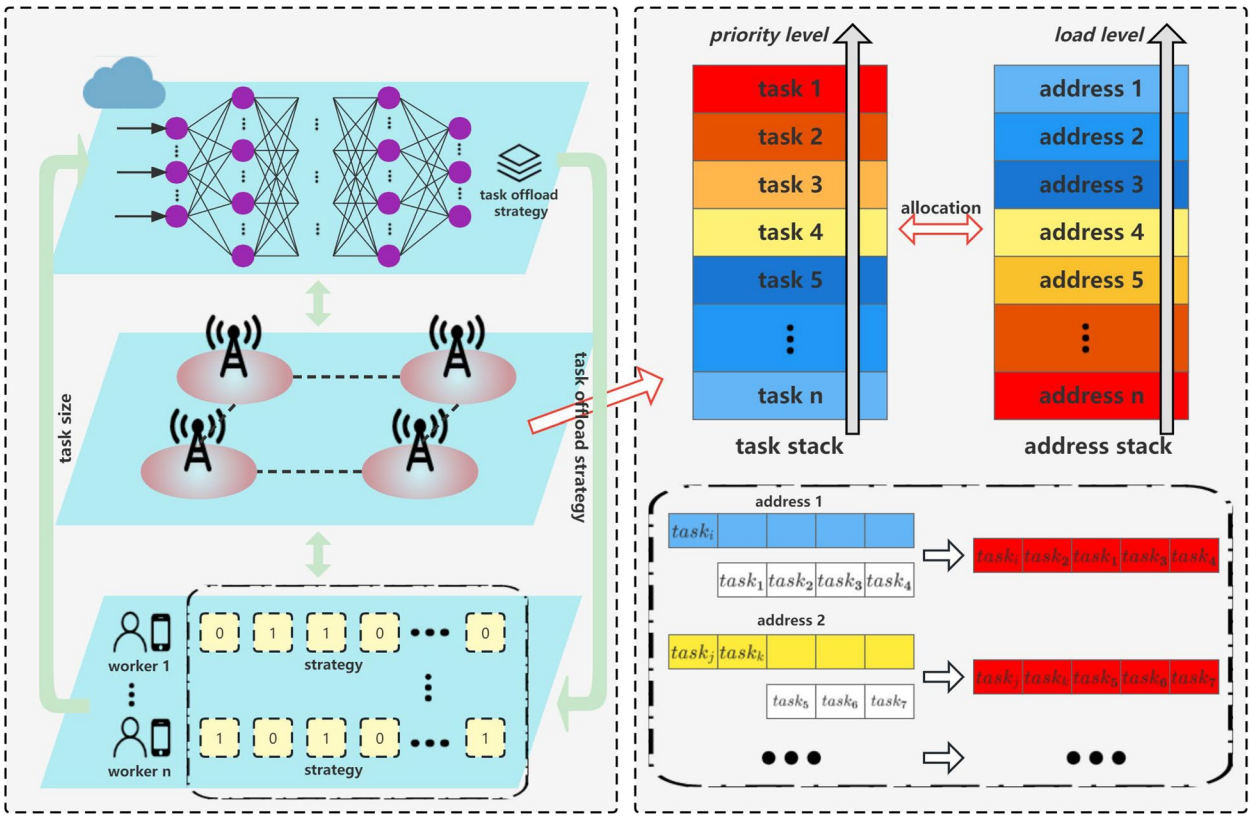

Fig. 1 System Model Design

DTOO algorithms

In this section, algorithms for generating offloading decisions using DNNs are presented. For the above-mentioned problem, we only need to find an optimal unloading strategy S_{cw}, so that W_{total} achieves the minimum value. Since the size of set S_{cw} is 2^{nm}, finding an optimal offloading strategy is NP-hard. Therefore, the approximate optimal unloading policy function Φ is obtained by training the DNN, such that $\Phi : S \leftarrow D$, as Eq. (9):

$$\arg\min\Phi(\tilde{S}) = \arg\min W_{total}(D, S_{c_i w_j}), \quad (i \in n; j \in m) \tag{9}$$

where D is the input of the model. The function value is optimized by performing a gradient descent algorithm during training by minimizing the cross-entropy loss function Eq. (10):

$$L(\Phi) = -\left[S\log\hat{S} + (1-S)\log\left(1-\hat{S}\right)\right] \tag{10}$$

The DTOO algorithm proposed in this paper is shown in Algorithm 1.

Input: $D = \{d_{c_1}, d_{c_2}, d_{c_1}, \ldots, td_{c_n}\}$
Output: S_{cw}
1. for d_{c_i} in D
2. input d_{c_i} to DNN
3. all DNNs generate candidate actions $\tilde{S}$
4. choose the optimal uninstall strategy $\arg\min\Phi(\tilde{S})$
5. gstore Φ at this time in memory
6. end for
7. select data from memory by sampling
8. sally out the next round of DNN training
9. update $L(\Phi)$
10. after training, output S_{cw}
11. return: S_{cw}

Algorithm 1 DTOO algorithm

Algorithm 1 shows the optimization process of task offloading strategy based on deep neural network. The input is the dataset D of the amount of data contained in the crowd task. The output is the approximate optimal task offloading strategy S_{cw}. Line1 traverses all task data volumes in D. Line2, Line3 Input d_{c_i} into all DNNs to get candidate offloading strategies S.

Line4-Line6 selects the optimal offloading strategy that minimizes $L(\Phi)$ and stores it in memory for the next round of training. Line7-Line9, sample data from memory for the next round of training, and update $L(\Phi)$. The training of Line10 and Line11 ends, and the approximate optimal unloading strategy S_{cw} is finally obtained.

Resource allocation algorithm

When crowd workers offload computing tasks to edge services, computing tasks are added to the stack in order based on priority. The administrator first matches the task at the top of the task stack with an address in the address stack with a high idle level. Then, when the address has fewer resources after a round of task allocation, the administrator will allocate the task to the next address with a higher idle level. The implementation process is shown in Algorithm 2.

Input: task offload strategy $S_{c_i w_j}$; edge servers set ES
Output: task stack; address stack
1. **for** $i = 1$ to n **do**
2. **for** $j = 1$ to m **do**
3. **if** $S_{c_i w_j} = 1$
4. add c_i to C_{edge}
5. **end for**
6. **end for**
7. sort by maximum allowable delay for task C_{edge}
8. $C_{level} \leftarrow C_{edge}$
9. Sort ES according to address idle level
10. $ES_{level} \leftarrow ES$
11. **for** $c_k in C_{edge}$
12. if $\sum_{u=1}^{k-1} t_{c_u} > c_k$
13. send c_k to other idle edge servers
14. **end for**
15. **return** task stack; address stack

Algorithm 2 Resource allocation algorithm

Algorithm 2 demonstrates the stack-based and sorted resource allocation process. The input is task offloading policy $S_{c_i w_j}$ and edge server set *ES*. The output is the task stack, the address stack. Line1-Line6 judges whether the task is offloaded to the edge server and obtains the task set C_{edge} that is offloaded to the edge server. Line7-Line8 prioritizes tasks according to their maximum allowable delay. Line9-Line10 sorts the addresses according to the idle class of edge servers. In Line11-Line14, when a new task is added, determine whether the total delay of the previous task is greater than that of the task, and if it is greater, assign the task to other idle addresses. Line15 returns the task stack and address stack for the next round of resource allocation.

Fig. 2 Mission information

Experiment and analysis

Dataset and experiment setup

The dataset used in the experiments is from the research-based general spatial crowdsourcing platform gMission [56]. As shown in Fig. 2, in the gMission dataset, each task contains its location, release time, and due date. The worker's data includes their location, and the time it takes to complete the task. Since the gMission dataset has a large period and a large amount of data, this paper selects 48 hours of data records from it. It contains 500 crowd workers, 1500 crowd tasks, and 29,654 worker check-in messages. The parameters used in the experiments include crowd workers, crowd tasks, and edge servers. In this paper, the number of tasks accepted by workers is set to be no more than three at most. The data processing speed of the mobile devices

Table 2 Experimental parameter settings

Variable	Value
r_{local}	$1.50 \times 10^7 bit/s$
q_{local}	$6.60 \times 10^8 J/bit$
x_{edge}	$1.25 \times 10^8 bit/s$
r_{edge}	$8.38 \times 10^8 bit/s$
y_{edge}	$7.81 \times 10^9 J/bit$
q_{edge}	$8.19 \times 10^9 bit/s$

carried by workers is $1.50 \times 10^7 bit/s$. The consumption of mobile devices processing data is $6.6 \times 10^8 J/bit$. In addition, this paper sets the number of edge servers within the task scope to five. The detailed parameter settings are shown in Table 2.

The experiments in this paper are all implemented in the Python environment. A laptop with Intel(R) Core (TM) i7-10750H CPU and 16GB memory was used.

Performance evaluation and comparative test

Figure 3 shows the convergence performance of the algorithm under different learning rates. From Fig. 3(a) and (b), it can be seen that when batch=64, the algorithm has the best convergence effect. In Fig. 3(c), the abscissa is the training step size, and the ordinate is the total resource consumption. It can be seen from Fig. 3 that the learning rate is too high or too low to achieve a good convergence effect. When the learning rate is 0.01, the convergence effect is better when 0.001. In Fig. 3(d), the abscissa is the training step size, and the ordinate is the gain rate. It can

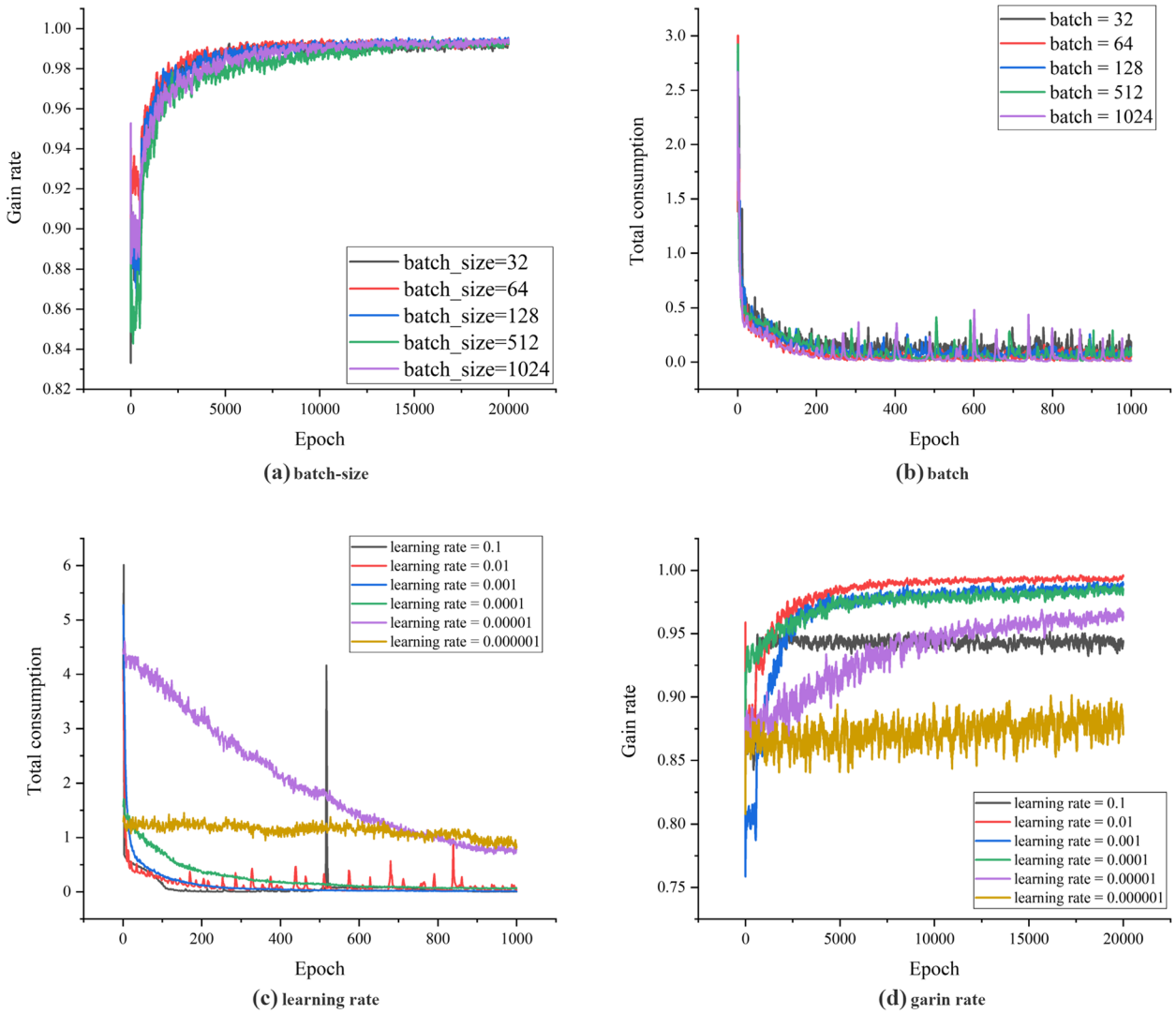

Fig. 3 Performance of the algorithm under different hyperparameters

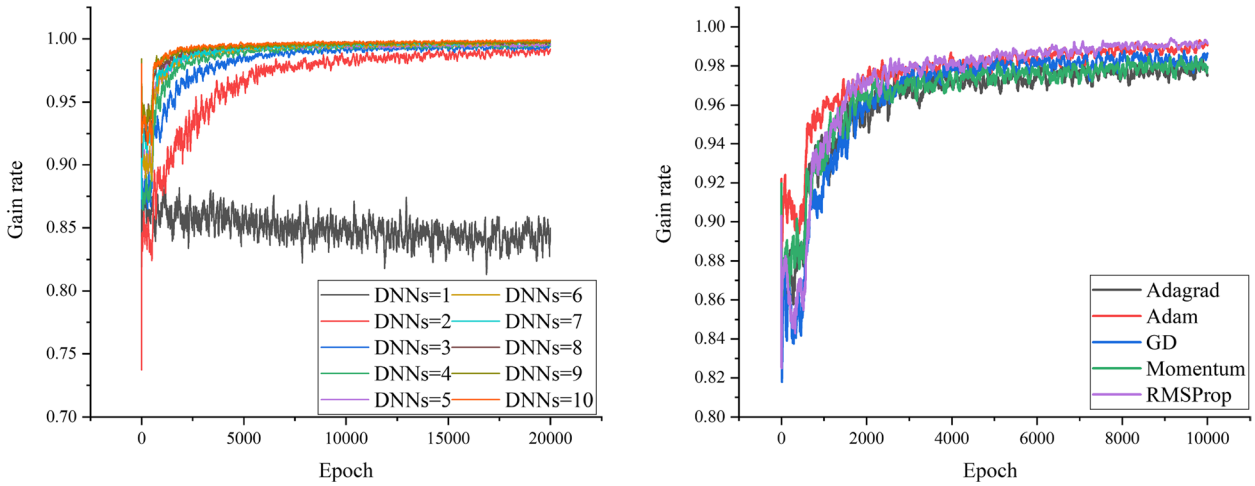

Fig. 4 Effect of optimizer and number of DNNs on performance

be seen from the figure that the best effect is when the learning rate is 0.01. Therefore, the batch is set to 64 and the learning rate is set to 0.01 in the experiments.

The effect of the number of DNNs on the convergence effect of the algorithm can be seen in Fig. 4(a). When there is only one DNN, the mutual learning between DNNs cannot be performed, so the algorithm cannot converge. When the number of DNNs is greater than 1, the gain rate increases as the number of DNNs increases, and the convergence requires fewer steps. Therefore, multiple DNNs can be selected for model training under the premise of hardware equipment. It can be seen in Fig. 4(b) that the convergence performance of the Adam optimizer is the best.

In order to verify the superiority of the proposed algorithm in terms of task completion rate, this paper conducts a comparison experiment with the RC [57] and LRU [58] algorithms and adds the traditional arrival time task ranking (ATR) algorithm for comparison. As shown in Fig. 5, the task completion rates of the four algorithms all decrease as the number of tasks increases, because when the number of tasks increases, the newly added tasks may not be allocated computing resources in time, resulting in task failure. It can be seen from Fig. 5 that the DTOO

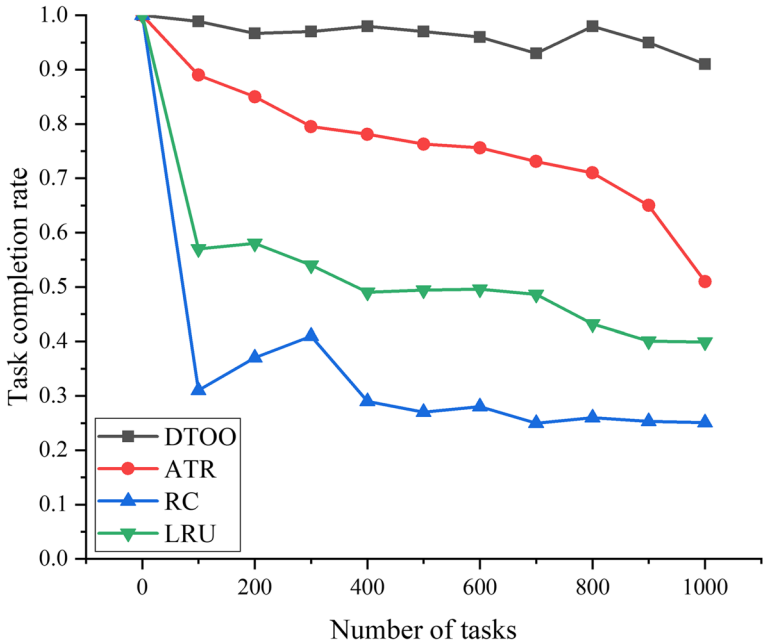

Fig. 5 Task completion rate under different number of tasks

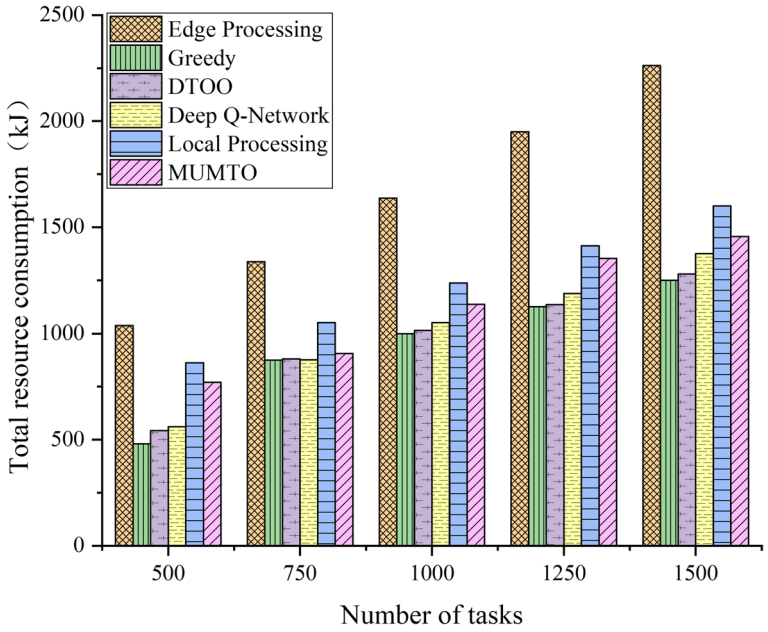

Fig. 6 Total resource consumption under different number of tasks

algorithm proposed in this paper has the highest task completion rate because in DTOO, tasks and edge server addresses are arranged in the stack according to their priorities so that tasks with higher priorities can obtain more computing resources, thus improving the task completion rate. It can be seen that the algorithm proposed in this paper is superior in terms of task completion rate.

In order to verify the superiority of the proposed algorithm in terms of resource consumption, this paper conducts comparison experiments with Deep Q-Network [45], MUMTO [59], Greedy algorithm, adds only local processing tasks and all tasks are offloaded to the Edge Server. As shown in Fig. 6, it is clear that the total resource consumption of all algorithms increases with the number of tasks. If all tasks are offloaded to edge servers for processing, high resource consumption will occur. The DTOO algorithm proposed in this paper is similar in utility to the Greedy algorithm, but since the Greedy algorithm will enumerate all the offloading strategies, it will occupy a considerable amount of system memory. Therefore, the algorithm proposed in this paper is superior in resource consumption.

Conclusion

This paper focuses on the task offloading problem of MCS in an edge cloud environment. Aiming at the problem of task offloading strategy selection, this paper proposed a DTOO algorithm based on DNN, which obtains an approximate optimal offloading strategy through learning among multiple neural units. It aims at solve the conflict between resource consumption and quality of service in practical MCS applications. To improve the completion rate of tasks, this paper proposed a stack-based resource sorting method. The tasks are arranged in the task stack according to their priority, and the server addresses are arranged in the address stack according to the idle level. After the task unloading is completed, according to the priority of the task for task allocation, thereby improving the task completion rate. Finally, performance tests and comparison experiments are carried out on the research-based general spatial crowdsourcing platform gMission dataset, and it is verified that the algorithm proposed in this paper has good performance in terms of balancing resource consumption and service quality as well as task completion rate. In the future work, the influence of workers' preferences on task offload decisions will be considered, and the task offload strategy will be further optimized.

Abbreviations

MCS	Mobile CrowdSourcing
DNN	Deep Neural Network
DTOO	Deep Neural Network-based Task Offloading Optimizatio
IoT	Internet of Things
EH	Energy Harvesting
MINLP	Mixed Integer Nonlinear Programming
GCGH	Gini Coefficient-based Greedy Heuristic
ATR	Arrival Time task Rankin
RC	tRend Caching
LRU	Least Recently Used
MUMTO	Multi-User Multi-Task Optimizatio

Author' contributions
M. L. wrote the main manuscript text . W. Y. modified and reviewed the paper. W. H. , T. X. , S. Z. and C. Z. participated in sorting out references. All authors reviewed the manuscript. The author(s) read and approved the final manuscript.

Authors' information
Lingkang Meng received his Bachelor degree in School of Computer and Control Engineering from Yantai University. He is a graduate student in School of Computer and Control Engineering at Yantai University. His research interests are mobile crowd sourcing and service computing.
Yingjie Wang received the received the Ph.D. degree in College of Computer Science and Technology from Harbin Engineering University. She visited Georgia State University from 2013/09 to 2014/09 as a visiting scholar. Dr. Wang is currently an Professor in the School of Computer and Control Engineering at Yantai University. She is a Postdoc in South China University of Technology. Her research interests are mobile crowdsourcing, privacy protection and service computing. She has published more than 60 papers in well known journals and conferences in her research field, which includes 2 ESI high cited papers. In addition, she has presided 2 National Natural Science Foundation of China project, 2 China Postdoctoral Science Foundation projects. Dr. Wang obtained the Shandong Province Artificial Intelligence Outstanding Youth Award.
Haipeng Wang received the Ph.D. degree from Naval Aviation University, in 2012, where he is currently an Professor. His research interests include the general area of intelligent perception and fusion, and big data technology and application. He also serves as a Reviewer for several distinguished journals, including IET RSN and IEEE AES.
Xiangrong Tong received the Ph.D. degree in School of Computer and Information Technology from Beijing Jiaotong University. Currently, he is a Full Professor of Yantai University. His research interests are computer science, intelligent information processing and social networks. He has published more than 50 papers in well known journals and conferences. In addition, he has presided and joined 3 national projects and 3 provincial projects.
Zice Sun received the Bechelor degree in the School of Computer and Control Engineering, Yantai University. He is currently pursuing the Master degree in the School of Computer and Control Engineering, Yantai University. His research interests are mobile crowdsourcing and blockchain.
Zhipeng Cai received his PhD and M.S. degrees in the Department of Computing Science at University of Alberta, and B.S. degree from Beijing Institute of Technology. Dr. Cai is currently an Professor in the Department of Computer Science at Georgia State University. Dr. Cai's research areas focus on Networking, Privacy and Big data. Dr. Cai is the recipient of an NSF CAREER Award. Dr. Cai is now a Steering Committee Co-Chair for WASA. He is an editor/guest editor for Algorithmica, Theoretical Computer Science, Journal of Combinatorial Optimization, IEEE/ACM Transactions on Computational Biology and Bioinformatics. He is a senior member of the IEEE.

Funding
This work was supported in part by the National Natural Science Foundation of China under Grant 62272405, the Youth Innovation Science and Technology Support Program of Shandong Provincial under Grant 2021KJ080, the Natural Science Foundation of Shandong Province, Grant ZR2022MF238, Yantai Science and Technology Innovation Development Plan Project under Grant 2021YT06000645, the Open Foundation of State key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) under Grant SKLNST-2022-1-12.

Availability of data and materials
The gMission dataset: http://gmission.github.io/.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 22 October 2022 Accepted: 25 April 2023
Published online: 11 May 2023

References

1. Wu Y, Zeng JR, Peng H, Chen H, Li C (2016) Survey on incentive mechanisms for crowd sensing. J Softw 27(8):2025–2047
2. Cai Z, He Z (2019) Trading private range counting over big IoT data. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE, Dallas, p 144–153
3. Zheng X, Cai Z (2020) Privacy-preserved data sharing towards multiple parties in industrial IoTs. IEEE J Sel Areas Commun 38:968–979
4. Xiang C, Zhou Y, Dai H, Qu Y, He S, Chen C, Yang P (2021) Reusing delivery drones for urban crowdsensing. IEEE Trans Mob Comput. https://doi.org/10.1109/TMC.2021.3127212
5. Lu Z, Wang Y, Li Y, Tong X, Mu C, Yu C (2021) Data-driven many-objective crowd worker selection for mobile crowdsourcing in industrial IoT. IEEE Trans Ind Inform. https://doi.org/10.1109/TII.2021.3076811
6. Sandhu AK (2021) Big data with cloud computing: Discussions and challenges. Big Data Min Anal 5:32–40
7. Duan Z, Li W, Zheng X, Cai Z (2019) Mutual-preference driven truthful auction mechanism in mobile crowdsensing. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE, Dallas, p 1233–1242
8. Hasenfratz D, Saukh O, Sturzenegger S, Thiele L et al (2012) Participatory air pollution monitoring using smartphones. Mob Sens 1:1–5
9. Brković M, Sretović V (2013) Smart solutions for urban development: potential for application in serbia. In: Congress Proceedings. Regional Development, Spatial Planning and Strategic Governance (RESPAG) 2nd International Scientific Conference, Belgrade. IAUS, Belgrade
10. Libelium (2017). http://www.libelium.com/. Accessed 2022
11. Wang Y, Cai Z, Tong X, Gao Y, Yin G (2018) Truthful incentive mechanism with location privacy-preserving for mobile crowdsourcing systems. Comput Netw 135:32–43
12. Qi L, Liu Y, Zhang Y, Xu X, Bilal M, Song H (2022) Privacy-aware point-of-interest category recommendation in internet of things. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2022.3181136
13. Li F, Wang Y, Gao Y, Tong X, Jiang N, Cai Z (2021) Three-party evolutionary game model of stakeholders in mobile crowdsourcing. IEEE Trans Comput Soc Syst. https://doi.org/10.1109/TCSS.2021.3135427
14. Chi C, Wang Y, Tong X, Siddula M, Cai Z (2021) Game theory in internet of things: A survey. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2021.3133669
15. Chen Y, Zhao J, Wu Y et al (2022) Qoe-aware decentralized task offloading and resource allocation for end-edge-cloud systems: A game-theoretical approach. IEEE Trans Mob Comput. https://doi.org/10.1109/TMC.2022.3223119
16. Xiang C, Yang P, Wu X, He H, Wang B, Liu Y (2015) istep: A step-aware sampling approach for diffusion profiling in mobile sensor networks. IEEE Trans Veh Technol 65:8616–8628
17. Chen Y, Gu W, Li K (2022) Dynamic task offloading for internet of things in mobile edge computing via deep reinforcement learning. Int J Commun Syst 5154
18. Kong L, Wang L, Gong W, Yan C, Duan Y, Qi L (2021) Lsh-aware multitype health data prediction with privacy preservation in edge environment. World Wide Web 1–16
19. Qi L, Lin W, Zhang X, Dou W, Xu X, Chen J (2022) A correlation graph based approach for personalized and compatible web apis recommendation in mobile app development. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2022.3168611
20. Qi L, Yang Y, Zhou X, Rafique W, Ma J (2021) Fast anomaly identification based on multi-aspect data streams for intelligent intrusion detection toward secure industry 4.0. IEEE Trans Ind Inform. https://doi.org/10.1109/TII.2021.3139363
21. Chen Y, Xing H, Ma Z, Chen X, Huang J (2022) Cost-efficient edge caching for noma-enabled IoT services. China Commun
22. Li K, Zhao J, Hu J et al (2022) Dynamic energy efficient task offloading and resource allocation for noma-enabled IoT in smart buildings and environment. Build Environ. https://doi.org/10.1016/j.buildenv.2022.109513

23. Huang J, Gao H, Wan S et al (2023) Aoi-aware energy control and computation offloading for industrial IoT. Futur Gener Comput Syst 139:29–37
24. Chen X, Jiao L, Li W, Fu X (2015) Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans Networking 24:2795–2808
25. 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. Digit Commun Netw 5:10–17
26. Bi R, Liu Q, Ren J, Tan G (2020) Utility aware offloading for mobile-edge computing. Tsinghua Sci Technol 26:239–250
27. Zhang Q, Wang Y, Yin G, Tong X, Sai AMVV, Cai Z (2022) Two-stage bilateral online priority assignment in spatio-temporal crowdsourcing. IEEE Trans Serv Comput 8:516–530
28. Wang Y, Cai Z, Zhan ZH, Zhao B, Tong X, Qi L (2020) Walrasian equilibrium-based multiobjective optimization for task allocation in mobile crowdsourcing. IEEE Trans Comput Soc Syst 7(4):1033–1046
29. Chen Y, Hu J, Zhao J, Min G (2023) Qos-aware computation offloading in leo satellite edge computing for IoT: A game-theoretical approach. Chin J Electron. https://doi.org/10.1109/TMC.2022.3223119
30. Shi W, Cao J, Zhang Q, Li Y, Xu L (2016) Edge computing: Vision and challenges. IEEE Internet Things J 3:637–646
31. Sun Z, Wang Y, Cai Z, Liu T, Tong X, Jiang N (2021) A two-stage privacy protection mechanism based on blockchain in mobile crowdsourcing. Int J Intell Syst (36-5). https://doi.org/10.1002/int.22371
32. Liu T, Wang Y, Li Y, Tong X (2020) Privacy protection based on stream cipher for spatio-temporal data in IoT. IEEE Internet Things J 7(9):7928–7940
33. Cai Z, Zheng X (2018) A private and efficient mechanism for data uploading in smart cyber-physical systems. IEEE Trans Netw Sci Eng 7(2):766–775
34. Wang T, Lu Y, Cao Z, Shu L, Zheng X, Liu A, Xie M (2019) When sensor-cloud meets mobile edge computing. Sensors 19(23):5324
35. Zhao W, Liu J, Guo H, Hara T (2018) Etc-IoT: Edge-node-assisted transmitting for the cloud-centric internet of things. IEEE Netw 32(3):101–107
36. Cai Z, Xiong Z, Xu H, Wang P, Li W, Pan Y (2021) Generative adversarial networks: A survey toward private and secure applications. ACM Comput Surv (CSUR) 54(6):1–38
37. Ren J, Yu G, He Y, Li GY (2019) Collaborative cloud and edge computing for latency minimization. IEEE Trans Veh Technol 68(5):5031–5044
38. Wang W, Wang Y, Duan P, Liu T, Tong X, Cai Z (2022) A triple real-time trajectory privacy protection mechanism based on edge computing and blockchain in mobile crowdsourcing. IEEE Trans Mob Comput 1–18
39. Xiang C, Zhang Z, Qu Y, Lu D, Fan X, Yang P, Wu F (2020) Edge computing-empowered large-scale traffic data recovery leveraging low-rank theory. IEEE Trans Netw Sci Eng 7(4):2205–2218
40. Xiang C, Li Y, Zhou Y, He S, Qu Y, Li Z, Gong L, Chen C (2022) A comparative approach to resurrecting the market of mod vehicular crowdsensing. In: Proc. IEEE Conf. Comput. Commun. IEEE, London, p 1–10
41. Xiang C, Yang P, Tian C, Zhang L, Lin H, Xiao F, Zhang M, Liu Y (2015) Carm: Crowd-sensing accurate outdoor rss maps with error-prone smartphone measurements. IEEE Trans Mob Comput 15(11):2669–2681
42. Wang Y, Gao Y, Li Y, Tong X (2020) A worker-selection incentive mechanism for optimizing platform-centric mobile crowdsourcing systems. Comput Netw 171(107):144
43. Dinh T, Tang J, La Q, Quek T (2017) Offloading in mobile edge computing: Task allocation and computational frequency scaling. IEEE Trans Commun 65(8):3571–3584
44. Wu H, Sun Y, Wolter K (2018) Energy-efficient decision making for mobile cloud offloading. IEEE Trans Cloud Comput 8(2):570–584
45. Xu J, Chen L, Zhou P (2018) Joint service caching and task offloading for mobile edge computing in dense networks. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, Honolulu, p 207–215
46. Cand ShuZ, Zhao Han Y, Min G, Duan H (2019) Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach. IEEE Internet Things J 7(3):1678–1689
47. Mao Y, Zhang J, Letaief K (2016) Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE J Sel Areas Commun 34(12):3590–3605
48. Zhao P, Tian H, Qin C, Nie G (2017) Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing. IEEE Access 5:11255–11268
49. Chen Y, Gu W, Xu J, et al (2022) Dynamic task offloading for digital twin-empowered mobile edge computing via deep reinforcement learning. China Commun. https://doi.org/10.1002/dac.5154
50. Mnih V, Kavukcuoglu K, Silver D, Rusu A, Veness J, Bellemare M, Graves A, Riedmiller M, Fidjeland A, Ostrovski G (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
51. Huang J, Wan J, Lv B, Ye Q et al (2023) Joint computation offloading and resource allocation for edge-cloud collaboration in internet of vehicles via deep reinforcement learning. IEEE Syst J. https://doi.org/10.1109/JSYST.2023.3249217
52. Xu Z, Wang Y, Tang J, Wang J, Gursoy MC (2017) A deep reinforcement learning based framework for power-efficient resource allocation in cloud rans. In: 2017 IEEE International Conference on Communications (ICC). IEEE, Paris, p 1–6
53. Ye H, Li G, Juang B (2017) Power of deep learning for channel estimation and signal detection in ofdm systems. IEEE Wirel Commun Lett 7(1):114–117
54. He Z Yand Zhang, Yu F, Zhao N, Yin H, Leung V, Zhang Y (2017) Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks. IEEE Trans Veh Technol 66(11):10433–10445
55. Huang L, Feng X, Qian L, Wu Y (2018) Deep reinforcement learning-based task offloading and resource allocation for mobile edge computing. In: International Conference on Machine Learning and Intelligent Communications. MLICOM, Hangzhou, p 33–42
56. gmission dataset. http://gmission.github.io/. Accessed 2022
57. Li S, Xu J, van der Schaar M, Li W (2016) Trend-aware video caching through online learning. IEEE Trans Multimed 18(12):2503–2516
58. Jin W, Li X, Yu Y, Wang Y (2013) Adaptive insertion and promotion policies based on least recently used replacement. IEICE Trans Inf Syst 96(1):124–128
59. Chen MH, Liang B, Dong M (2016) Joint offloading decision and resource allocation for multi-user multi-task mobile cloud. In: 2016 IEEE International Conference on Communications (ICC). IEEE, Paris, p 1–6

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen journal and benefit from:

- ► Convenient online submission
- ► Rigorous peer review
- ► Open access: articles freely available online
- ► High visibility within the field
- ► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com