Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

PPDO: a privacy-preservation-aware delay optimization task-offloading algorithm for collaborative edge computing*

Chao JING[†‡1,2], Jianwu XU[1]

[1]*College of Computer Science and Engineering, Guilin University of Technology, Guilin 541004, China*

[2]*Guangxi Key Laboratory of Embedded Technology and Intelligent System, Guilin 541004, China*

[†]E-mail: jingchao@glut.edu.cn

Received Oct. 31, 2023; Revision accepted Jan. 5, 2024; Crosschecked Nov. 13, 2024

Abstract: Although collaborative edge computing (CEC) systems are beneficial in enhancing the performance of mobile edge computing (MEC), the issue of user privacy leakage becomes prominent during task offloading. To address this issue, we design a privacy-preservation-aware delay optimization task-offloading algorithm (PPDO) in a CEC system. By considering location and usage pattern privacy protection, we establish a privacy task model to interfere with the edge server and ensure user privacy. To address the extra delay arising from privacy protection, we subsequently leverage a Markov decision processing (MDP) policy-iteration-based algorithm to minimize delays without compromising privacy. To simultaneously accelerate the MDP operation, we develop an extension that improves the PPDO by optimizing the action set. Finally, a comprehensive simulation was conducted using the edge user allocation (EUA) dataset. The results demonstrated that PPDO achieves an optimal trade-off between privacy protection and delay with a minimum delay compared with existing algorithms. Moreover, we examined the advantages and disadvantages of improving PPDO.

Key words: Collaborative edge computing; Task offloading; Privacy protection; Markov decision process
https://doi.org/10.1631/FITEE.2300741 **CLC number:** TP393

1 Introduction

With the rapid growth of the scale of mobile networks, intelligence-based mobile devices have become widely used in numerous fields. Owing to the many requirements of resource-intensive applications (Ouyang et al., 2019), there is a significant increase in data volume and demand for less delay (Wang F et al., 2018), especially for applications that require a high quality of service (QoS) and are vulnerable to delays (Taleb et al., 2017; Chen et al., 2020), such as mobile gaming (Yang et al., 2020), virtual reality (VR) (Lin et al., 2021), image video processing (Mach and Becvar, 2017), and speech recognition (Yousaf et al., 2018). However, the resources of mobile cloud computing (MCC) are usually located far from the mobile devices, with high latency and energy consumption for data transmission owing to network congestion and speed limitations (Gao et al., 2023a). Fortunately, these issues can be addressed by the emerging technology of mobile edge computing (MEC) (Abbas et al., 2018). By situating servers at the edge of the network, edge computing can minimize latency (Dong et al., 2023).

Collaborative edge computing (CEC) systems have attracted significant attention owing to the

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 62362018), the Guangxi Key Research and Development Program (Nos. GUIKEAB23075116 and GUIKEAB23075175), and the Innovation Project of Guangxi Graduate Education (No. YCSW2023350)

ORCID: Chao JING, https://orcid.org/0000-0002-4695-8746

constrained edge server (ES) resources of MEC as compared to MCC (Lee et al., 2021). CEC, characterized by the distribution of computational tasks among distributed edge nodes (Sahni et al., 2019), ensures appropriate collaboration on ESs when faced with a risk of privacy leakage owing to a reduced communication overhead in selecting a closer ES. Moreover, when the network conditions are poor or the ES processing queues are congested, certain tasks can be processed by mobile devices instead of being offloaded to the ES. By appropriately offloading tasks, CEC can greatly improve the performance of MEC. Several studies have addressed the costs associated with task offloading in edge computing, generally including execution delay and energy consumption. Mao YY et al. (2017) proposed an algorithm based on Lyapunov optimization to solve the cost optimization problem in edge computing systems with limited computational resources. Similarly, Cao et al. (2021) adopted the Lyapunov optimization framework to minimize user QoS losses stemming from network delay and low frame rates under cost constraints. Chu et al. (2023) proposed a distributed online version of the mechanism to maximize user QoS in MEC by jointly optimizing service caching, resource allocation, and task offloading decisions. Qin et al. (2021) devised a threshold-based distributed task offloading algorithm that allows MEC users to update their thresholds based on their cost functions. Ren et al. (2019) considered collaboration between cloud and edge computing, with a joint communication resource and computation resource allocation problem formulated to minimize the latency of mobile devices. Wang JD et al. (2021) designed a resource allocation scheme that rationally assigns computational and network resources under changeable edge computing conditions. However, most of these studies focused on cost optimization, ignoring the importance of privacy protection during offloading.

When ESs are located closer to mobile devices, the QoS of users can benefit from the advantages of edge computing. However, edge computing raises the severe issue of privacy leakage. In certain cases, ESs and mobile edge computing environments cannot be trusted. For example, the private information of users may be exposed during task offloading (He XF et al., 2017; Wang ZB et al., 2019; Qian et al., 2021; Wang TS et al., 2021). This creates an opportunity for malicious users to leverage private information for tracking, fraud, attacks, and illegal exploitation (He T et al., 2017; He XF et al., 2020; Mao S et al., 2022). The process of task offloading has previously been associated with privacy leaks (Bai et al., 2020). Because MEC servers may attempt to obtain user locations and usage patterns (Min et al., 2019), it is essential to determine the trade-off between latency optimization and privacy protection.

Numerous studies have addressed the privacy protection issues inherent to edge computing. One recent study (Wang ZB et al., 2023) advocated a privacy-preservation-aware task-offloading framework based on a location perturbation scheme that provides differential privacy guarantees. Gao et al. (2023b) introduced a deep reinforcement learning method to minimize the risk of location privacy leakage during task offloading. Another study (Wang WX et al., 2020) employed a location-privacy-preservation-aware migration framework and defined the total system cost as a combination of the risk of location privacy leakage and delay of task offloading. Hua et al. (2023) computed risk associated with location leakage and applied it as a constraint in task offloading. In Zhao et al. (2023), privacy-preserving models were designed by hiding user offloading decisions and intentionally offloading redundant tasks. Nonetheless, most of the aforementioned studies were concerned only with protecting location privacy while ignoring usage pattern. Although Zhao et al. (2023) protected both location and usage pattern privacy by offloading redundant tasks, these tasks were derived from the historical tasks of the user device that extended the latency.

In contrast to the aforementioned methods, we propose a privacy-preservation-aware offloading algorithm in CEC (PPDO) with the objective of optimizing delay under privacy considerations. In the CEC framework, tasks associated with mobile users can be offloaded to the edge server or processed by a local device. To ensure privacy during offloading, we enhance the protection of both location and usage pattern by integrating the concept of privacy tasks. The main function of a privacy task is to create interference wherein the ES cannot analyze the user's location or usage pattern information. Although the traditional Markov decision processing (MDP) based approach (Ksentini et al., 2014) effectively optimizes delays during task offloading, the direct use of MDP is time-consuming owing to the selection of optimal

actions. We therefore improve MDP by eliminating the redundant action set. Thus, the proposed algorithm efficiently optimizes latency when tasks are offloaded, achieving a balance between privacy protection and latency optimization. The main contributions of this work are summarized as follows:

1. We establish a multi-user CEC that minimizes the delay of task offloading without compromising privacy preservation. Specifically, we enhance the privacy protection scheme with respect to location and usage pattern. We also define privacy tasks to interfere with ES operations to prevent privacy leaks.

2. We develop an improved MDP-based privacy-preservation-aware task-offloading algorithm in CEC under the enhanced privacy protection scheme. The improved MDP model embraces a state set, probabilistic transfer function, reward function, and action set. Because the policy iteration of MDP may be highly time-intensive, we reduce the exploration of the action set. Specifically, given a sufficiently small number of tasks, or a sufficiently large CPU processing capability of the ES, our algorithm bypasses the local execution of the task in the action set. Moreover, given a sufficient CPU processing capability of the local device, ESs situated far away from the user are omitted from the action set. Combining these principles with the privacy protection scheme, the proposed algorithm successfully minimizes the latency.

3. We conduct a simulation using the edge user allocation (EUA) dataset (Lai et al., 2018). Several competitive algorithms are selected for comparative purposes, including the random offloading algorithm, policy-iteration-based algorithm (Wang WX et al., 2020), and genetic algorithm (Wang ZB et al., 2023). The results demonstrate that the proposed algorithm outperforms comparative algorithms in minimizing delay under the consideration of privacy preservation. Subsequently, we further discuss the potential improvements of PPDO.

2 Problem formulation

2.1 System model

Fig. 1 depicts the overall model of a CEC system. Let $m = 1, 2, ..., M$ and $n = 1, 2, ..., N$ denote the numbers of mobile users (MUs) and ESs, respectively. Each mobile terminal generates multiple computing tasks $k = 1, 2, ..., K$, where each attribute of a task consists of a tuple $\{c_k, s_k\}$. Here s_k is the

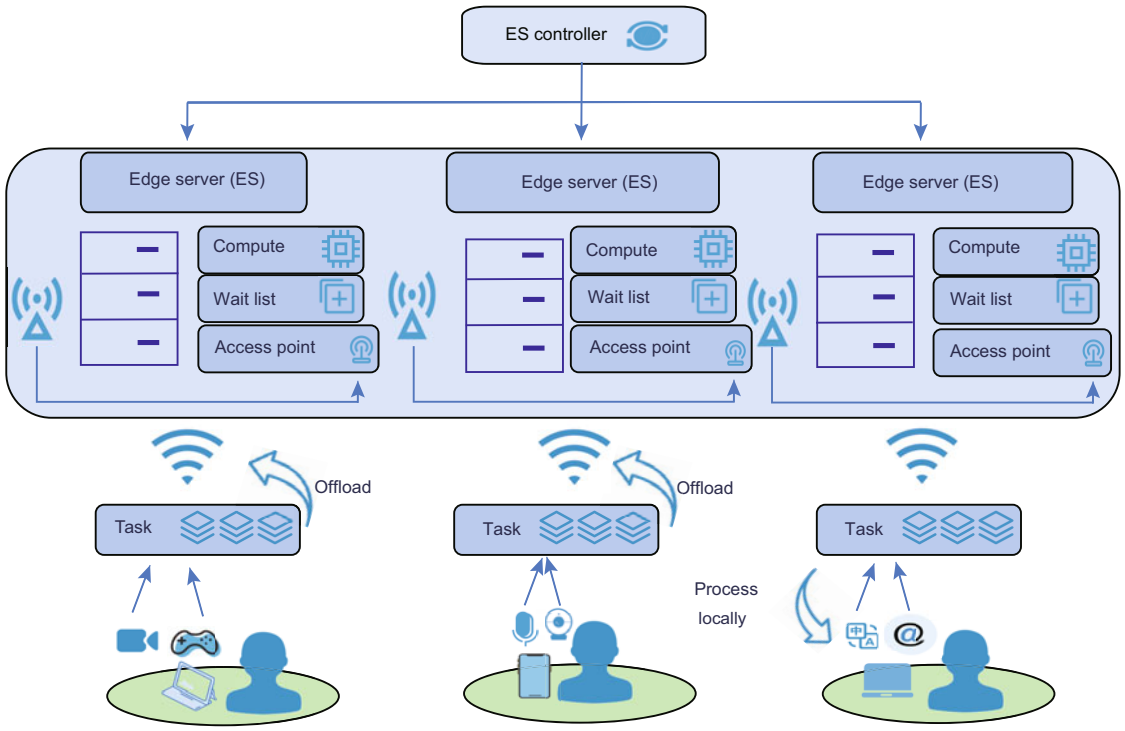

Fig. 1 Illustration of the collaborative edge computing system model with multiple users

task's data volume, which remains constant during task processing, and c_k is the number of CPU cycles required for the task. Letting δ_k be the number of cycles per data unit, c_k can be computed as $s_k\delta_k$. The MU connects to the ES through the network and uses a binary offloading mode; thus, each task can be processed only locally by the MU or offloaded to an ES.

2.1.1 Local computing model

Tasks generated by terminal devices of the MU in CEC can be divided into local execution and offload to ESs. However, tasks cannot be split during execution.

The set of offloading decision variables for the task is $X = \{x_{i,k,j}|i = 1,2,...,M,\ j = 1,2,...,N,\ k = 1,2,...,K\}$, where $x_{i,k,j} \in \{0,1\}$. If $x_{i,k,j} = 0$, task k is processed at local device i; otherwise, it is offloaded to ES j for processing. The notations used are listed in Table 1.

Table 1 Notations used in this paper

Parameter	Definition
m	Number of MUs, $m = 1,2,...,M$
n	Number of ESs, $n = 1,2,...,N$
k	Number of tasks, $k = 1,2,...,K$
f^{local}	Local CPU capability
s^{local}	Mobile device storage capacity
f^{ES}	ES CPU capability
s^{ES}	ES storage capacity
s_k	Data size of task k
c_k	CPU cycle number of task k
δ_k	CPU cycle number per unit data size of task k
d_{ij}	Distance between user device i and ES j
s_i^{privacy}	Data size of privacy task from user i
c_i^{privacy}	CPU cycle number of privacy task from user i
$\delta_i^{\text{privacy}}$	CPU cycle number per unit data size of privacy task from user i
X	Offloading decision
B_{w}	Bandwidth
p	Transmission power
σ	Noise power
A_{s}	Small-scale attenuation factor
B	Channel fading factor
h_{ij}	Channel gain between user device i and ES j

MU: mobile user; ES: edge server

Locally executed tasks are subject only to delays from computation and queuing, as no transmission delay is incurred. The CPU frequency of the local device is f_i^{local} $(i = 1,2,...,M)$ and the incurred storage capacity is s_i^{local}. s_{ik} represents the size of data with task k from user device i and c_{ik} denotes the number of cycles required by task k on local device i, so the delay incurred by local execution can be derived from

$$\begin{aligned} t_i^{\text{local}} &= \frac{1}{f_i^{\text{local}}}\sum_{k=1}^{K} c_{ik} \\ \text{s.t. } &\sum_{k=1}^{K}(1 - x_{i,k,j})s_{ik} \le s_i^{\text{local}}. \end{aligned} \tag{1}$$

The constraint indicates that the sum of locally executed task data is no larger than the device storage capacity. When this threshold is exceeded, any subsequent tasks must be queued. The queuing delay is the sum of the execution delays from the previous task $\bar{k}$ on local device i where $\bar{k} \neq k$, calculated by

$$t_i^{\text{local\_queue}} = \frac{1}{f_i^{\text{local}}}\sum_{\bar{k}=1,\bar{k}\neq k}^{K} c_{i\bar{k}}. \tag{2}$$

Therefore, the total delay in local task execution can be computed by

$$t_{\text{sum}}^{\text{local}} = \sum_{i=1}^{M}\left(t_i^{\text{local}} + t_i^{\text{local\_queue}}\right). \tag{3}$$

2.1.2 Edge server computing model

User devices are connected to the ES through the network, and the uploading processes of tasks are considered during network transmission. In the model specifications, B_{w} denotes the channel bandwidth of the network transmission, p indicates the transmission power, σ denotes the noise power, A_{s} is used as a small-scale attenuation factor, B represents the channel fading factor, and d_{ij} denotes the communication distance between user device i and ES j. Thus, the channel gain between user device i and ES j can be computed using

$$h_{ij} = A_{\text{s}}d_{ij}^{-B}. \tag{4}$$

Let $ph_{i'j}$ represent the interference generated by other MUs (except MU i) to ES j. Based on the Shannon formula, the transmission rate from user device i to ES j can be calculated by

$$v_{ij} = B_{\text{w}}\log_2\left(1 + \frac{ph_{ij}}{\sum\limits_{i'=1,2,...,M,i'\neq i} ph_{i'j} + \sigma^2}\right). \tag{5}$$

Therefore, if the tasks generated by the user device are offloaded to a certain ES j, the task data must be transmitted to the server through the network. s_{kj} is the size of task k data to ES j, and transmission delay is the accumulation of the selected device users to ES j, defined by

$$t_j^{\text{tran}} = \sum_{k=1}^{K} x_{i,k,j} \frac{s_{kj}}{v_{ij}},\ i = 1, 2, ..., M. \tag{6}$$

Following the data transfer, the ES begins to compute the task. f_j^{ES} and s_j^{ES} $(j = 1, 2, ..., N)$ denote the processing and storage capacities of the ES, respectively. The computational delay of the task offloaded to the ES can be obtained by

$$\begin{aligned} t_j^{\text{ES}} &= \frac{1}{f_j} \sum_{k=1}^{K} c_{kj} \\ \text{s.t.} \sum_{k=1}^{K} x_{i,k,j} s_{kj} &\le s_j^{\text{ES}},\ i = 1, 2, ..., M, \end{aligned} \tag{7}$$

where s_{kj} denotes the size of task k data to ES j, and c_{kj} is the number of cycles required by task k to ES j.

When the amount of task data offloaded to the ES exceeds its current storage capacity, any forthcoming tasks $\bar{k}$ must be queued. The queuing delay, represented by the sum of the task execution delays of other users who have already entered the processing queue, is obtained by

$$t_j^{\text{ES\_queue}} = \frac{1}{f_j} \sum_{\bar{k}=1, \bar{k} \neq k}^{K} c_{\bar{k}j}. \tag{8}$$

The total delay from task offloading to ES execution can then be calculated as

$$t_{\text{sum}}^{\text{ES}} = \sum_{j=1}^{N} \left(t_j^{\text{tran}} + t_j^{\text{ES}} + t_j^{\text{ES\_queue}} \right). \tag{9}$$

2.1.3 Privacy information

1. Risk of privacy leakage

The privacy of location relates to the distance between users and servers. Generally, for a larger distance, the risk of user's location privacy leakage will be lower (Hua et al., 2023). According to Wang WX et al. (2020), we can obtain the risk of user's location privacy leakage by

$$l = \begin{cases} 0, & d_{ij} > d_{\text{thr}}, \\ a_{\text{l}} + b_{\text{l}} h^{d_{ij}}, & d_{ij} \le d_{\text{thr}}, \end{cases} \tag{10}$$

where a_{l}, b_{l}, and h are constants. According to Eq. (10), l relates to d_{ij} and d_{thr}. To highlight these key factors, we reformulate Eq. (10) by integrating $\frac{d_{ij}}{d_{\text{thr}}}$; thus, the risk of location privacy leakage can be redefined as

$$l_{\text{l}} = \begin{cases} 0, & d_{ij} > d_{\text{thr}}, \\ a_{\text{l}} + b_{\text{l}} h^{\frac{d_{ij}}{d_{\text{thr}}}}, & d_{ij} \le d_{\text{thr}}. \end{cases} \tag{11}$$

In CEC, it is equally important to protect usage pattern privacy. Users exhibit different usage patterns when running applications on their devices (Zhao et al., 2018). According to the previous equations, for users who strictly satisfy $d_{ij} > d_{\text{thr}}$, their location privacy risk is 0. It means that there is no location privacy leakage problem. For these users, it is imperative to prioritize the privacy of usage pattern. Accordingly, the risk of usage pattern privacy can be defined by

$$l_{\text{u}} = \begin{cases} a_{\text{l}} + b_{\text{l}} h^{\frac{d_{ij}}{d_{\text{thr}}}}, & d_{ij} > d_{\text{thr}}, \\ 0, & d_{ij} \le d_{\text{thr}}. \end{cases} \tag{12}$$

Conversely, we prioritize location privacy for any users satisfying $d_{ij} \le d_{\text{thr}}$.

2. Privacy task

In our framework, all privacy tasks generated are offloaded. The number of privacy tasks is determined by the privacy risks l_{l} and l_{u}. Each privacy task has the same attributes (e.g., data size and number of cycles) as a normal task on the user device; however, the value of each attribute is less than that in a normal task. Because each privacy task interferes with the ES, higher attribute values may significantly impact delay.

To protect location privacy, mobile devices deliberately offload tasks under poor network conditions (Zhao et al., 2023). Therefore, the user device is added to the privacy task queue, and the tasks are offloaded to the previous ES when the user switches service. To protect the privacy of usage pattern, privacy tasks are added to the user's normal task queue to perturb the presumption of the ES regarding the usage pattern. Fig. 2 illustrates the process of privacy task offloading. The specific privacy-preserving scheme is described in Section 3.2.

Let s^{privacy} and c^{privacy} denote the data size and number of cycles required by the privacy task, respectively. The transmission delay of the privacy

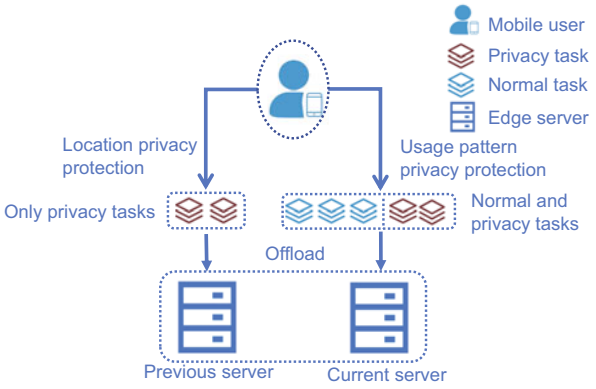

Fig. 2 Illustration of the privacy task

from selected user devices to ES j is

$$t_j^{\text{privacy\_tran}} = \sum_{i=1}^{M} \frac{s_{ij}^{\text{privacy}}}{v_{ij}}. \tag{13}$$

All privacy tasks are offloaded to the ES for execution, and the processing delay for a privacy task from user device i to ES j is defined by

$$\begin{aligned} t_j^{\text{privacy}} &= \frac{1}{f_j} \sum_{i=1}^{M} c_{ij}^{\text{privacy}} \\ \text{s.t.} \sum_{i=1}^{M} s_{ij}^{\text{privacy}} &\le s_j^{\text{ES}}, \end{aligned} \tag{14}$$

where c_{ij}^{privacy} indicates the number of cycles required by the privacy task from device user i to ES j.

When the volume of the task data received by the ES exceeds the current storage capacity, privacy task processing from other users $\bar{i}$ must be queued, where $c_{\bar{i}j}^{\text{privacy}}$ represents the number of cycles required by the privacy task from other users $\bar{i}$ (except i) to ES j:

$$t_j^{\text{privacy\_queue}} = \frac{1}{f_j} \sum_{\bar{i}=1, \bar{i} \neq i}^{M} c_{\bar{i}j}^{\text{privacy}}. \tag{15}$$

The cumulative delay incurred by privacy tasks is obtained by

$$\begin{aligned} t_{\text{sum}}^{\text{privacy}} = \sum_{j=1}^{N} \Big(& t_j^{\text{privacy\_tran}} + t_j^{\text{privacy}} \\ & + t_j^{\text{privacy\_queue}} \Big). \end{aligned} \tag{16}$$

2.2 Problem statement

Combining the established CEC model with the processing models for normal and privacy tasks, the total delay becomes the sum of all task delays, including those associated with execution processes and privacy tasks. The objective is to minimize the total delay, as defined by

$$\begin{aligned} \text{P1}: \min T_{\text{sum}} &= t_{\text{sum}}^{\text{local}} + t_{\text{sum}}^{\text{ES}} + t_{\text{sum}}^{\text{privacy}} \\ \text{s.t.} \sum_{k=1}^{K} (1 - x_{i,k,j}) s_{ik} &\le s_i^{\text{local}}, \\ \sum_{k=1}^{K} x_{i,k,j} s_{kj} &\le s_j^{\text{ES}}. \end{aligned} \tag{17}$$

The first constraint denotes the tasks to be executed locally while the local capacity is sufficient. The second constraint indicates the tasks to be offloaded to a certain ES if it has sufficient capacity.

3 Algorithm design

3.1 Overview

The following subsections describe the algorithms for integrating privacy protection and latency optimization in multi-user CEC systems. The privacy protection component offloads privacy tasks encompassing location and usage pattern privacy, whereas the proposed policy-iteration algorithm optimizes latency.

3.2 Privacy-task-based privacy preservation

To protect the users' location and usage pattern privacy during task offloading in CEC, we implement privacy tasks, which are offloaded to interfere with the ES responsible for task execution, inhibiting the attacker from analyzing privacy information. Fig. 3 illustrates the risk of location and usage pattern privacy leakage.

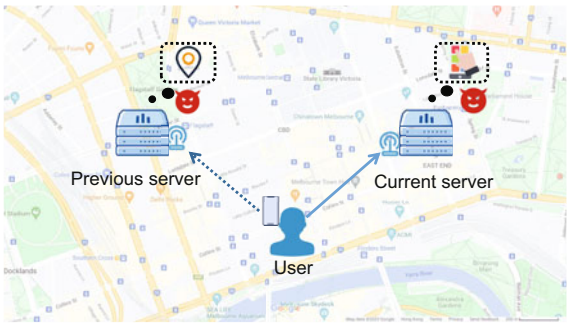

Fig. 3 Location and usage pattern privacy leakage risks

1. Location privacy protection. ESs can infer user location information through offloaded tasks, as mobile users in MEC may switch servers or execute tasks locally under poor network conditions (Min et al., 2019; Bai et al., 2020). To solve this problem, our algorithm generates a list of all privacy tasks, with their number being determined by the privacy leakage risk l_{l} and number of normal tasks u_k, denoted as $\lfloor l_{\mathrm{l}} u_k \rfloor$. Consequently, the ES continues to function as if it was receiving offloaded tasks from the user, and therefore cannot infer the user's location information. However, the user is subject to the delay stemming from the privacy tasks.

2. Usage pattern privacy protection. The leakage of usage pattern privacy can be primarily attributed to the ES evaluating data associated with offloaded tasks (Min et al., 2019; Zhao et al., 2023). To protect the privacy of the usage pattern, we add privacy tasks to the task list offloaded to the ES, with the number of privacy tasks being denoted as $\lfloor l_{\mathrm{u}} u_k \rfloor$. These privacy tasks interfere with the ES during the evaluation of task data, thereby protecting the usage pattern. Similarly, the user is subject to delay stemming from the privacy tasks.

Fig. 4 illustrates a general privacy protection procedure for mobile users. Although both types of privacy are preserved by offloading privacy tasks, the specific details differ between the two. The process is described in Algorithm 1.

3.3 MDP-based offloading algorithm

The following subsections describe the integration of the transformed CEC model into an MDP model, as well as problem design under this model, followed by solving the latency optimization problem stemming from task execution using the policy iteration algorithm.

3.3.1 MDP model

An MDP model typically includes a state set, an action set, the system transfer probability, and a reward function. We use an iteration strategy to determine the best value function that solves the MDP problem and obtains the user's offloading decision.

For the state set, the state space is defined as $S = \{s | s = (d, q)\}$, where d denotes the distance between the user device and each ES, and q denotes the task processing queue of the ES. The CEC system defines actions associated with the users' selection of ES nodes, with the action set being defined as $A = \{a_1, a_2, ..., a_M\}$. For example, a_i $(i = 1, 2, ..., M)$ indicates that user i offloads a task to a certain ES. Each policy corresponds to an action in action set A, denoted by $\pi(s) \in A(s)$.

The state transfer probability of the CEC system is related to distance d and queue q in the state set. The distance transfer function is shown as

$$P(d|d', a) = \frac{c_1}{nd'}, \tag{18}$$

where $a \in A$, n is the number of ESs, d' denotes the

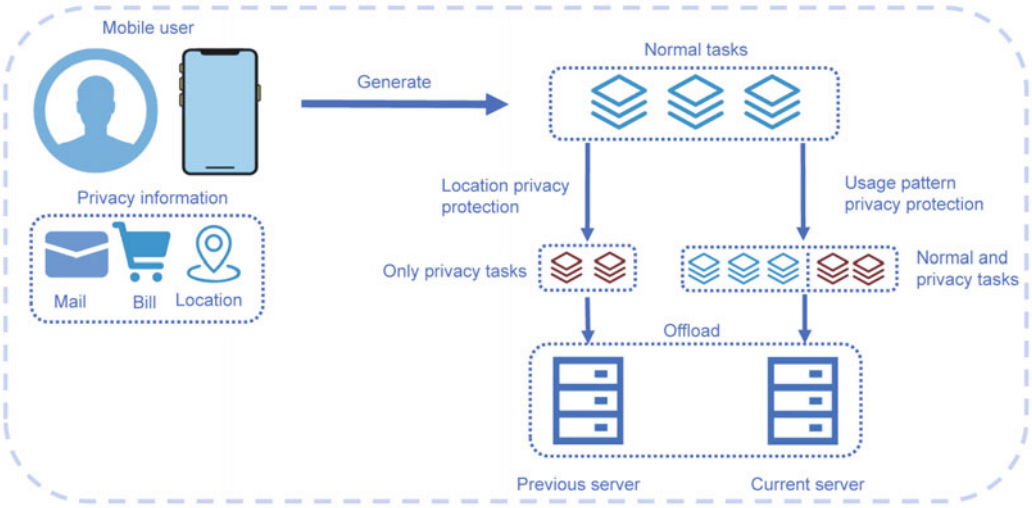

Fig. 4 Illustration of a general procedure on privacy-task-based privacy preservation

Algorithm 1 Privacy-task-based privacy-preservation-aware scheme

Initialize privacy leakage risk l_{l}, l_{u}, and task number u_k for all $i = 1, 2, ..., M$
/* Location privacy preservation */
loop
  for all $i = 1, 2, ..., M$ **do**
    if $l_{\mathrm{l}} > 0$ **then**
      Generate $\lfloor l_{\mathrm{l}} u_k \rfloor$ privacy tasks
      Set the new task list by $\lfloor l_{\mathrm{l}} u_k \rfloor$ privacy tasks
    end if
  end for
  if user device i switches the ES **then**
    Offload privacy tasks to the previous ES
  end if
end loop
/* Usage pattern privacy preservation */
loop
  for all $i = 1, 2, ..., M$ **do**
    if $l_{\mathrm{u}} > 0$ **then**
      Generate $\lfloor l_{\mathrm{u}} u_k \rfloor$ privacy tasks
      Add $\lfloor l_{\mathrm{u}} u_k \rfloor$ privacy tasks to the task list
      Offload the task list to the current ES
    end if
  end for
end loop

distance between the ES to be selected and the user device, and c_1 is a constant. For CEC systems with a fixed number of ESs, a smaller distance between the user device and ES indicates a higher transfer probability.

The queue transfer function, which relates to the queue state of the edge device, is shown as

$$P(q|q', a) = \frac{c_2}{q_{\mathrm{t}}}, \tag{19}$$

where q_{t} is the number of tasks queued by the edge device and c_2 is a constant. Similarly, a smaller number of tasks in the queue ensures a greater probability of queue transfer. Letting s' be the state immediately following state s and d' be equivalent to distance d between the user device and ES, if d and the transfer probability of the task processing queue q of the ES are mutually independent, the transfer probability of the system is obtained by

$$P(s|s', a) = P(d|d', a)P(q|q', a). \tag{20}$$

In the MDP model developed in this study, the reward function is defined as the weighted sum of the time delays generated following the selection of an ES expressed as

$$r = \omega t_{\mathrm{sum}}, \tag{21}$$

where $\omega > 0$ is a weighting factor that takes values within [1, 10] and t_{sum} is the total delay incurred by task execution.

In summary, the value function of the task under the MDP model is formulated as

$$V(s, a) = r + \gamma \sum_{s'} P(s|s', a)V(s', a), \tag{22}$$

where $\gamma \in [0, 1]$ is the discount factor that indicates the degree of influence of the future state on the current state. Because reward r is proportional to the task delay, an optimal solution to this problem must minimize the value function:

$$\mathrm{P2}: \min_{a \in A, s \in S} V(s, a). \tag{23}$$

Observing P2, for the transformed MDP problem, an algorithm must be used to solve the minimum value function $V(s, a)$, which converges to policy $\pi(s)$ and then remains constant, and its action $a(s)$ is the node identified as the optimal offloading decision for the user in the CEC.

3.3.2 Policy-iteration-based privacy-preservation-aware offloading algorithm

In this subsection, we propose the PPDO privacy-preservation-aware algorithm, which generates offloading decisions for users with the objective of reducing the latency of task offloading. We identify the delay incurred by user task offloading and privacy protection in CEC systems. The algorithm first initializes the selection of value function $V(s)$ and policy $\pi(s)$ for each user in the context of conversion to MDP, using a random policy for the initial action selection, and reward r is used in the value function for the initial action in the initialization phase. Subsequently, the policy evaluation phase initiates to compute value function $V(s)$ for a given strategy with the objective of making $V(s)$ converge. Finally, the policy improvement phase focuses mainly on the selection of $\pi(s)$ based on the difference in $V(s)$. Because r is proportional to the total delay of task offloading in this study, smaller $V(s)$ is required in this phase. An optimal policy is generated when $\pi(s)$ no longer changes in this phase, and the algorithm terminates. The overall process is presented in Algorithm 2.

Algorithm 2 Privacy-preservation-aware delay optimization task-offloading algorithm

Input: state set S, action set A, and probability set P
Output: optimal strategy $\pi(s)$
Randomly set $\pi(s)$ and $V(s)$ for all $s \in S$
/* Policy evaluation */
loop
  $\Delta \leftarrow 0$
  for each $s \in S$ **do**
    $v \leftarrow V(s)$
    $V(s,a) \leftarrow r + \gamma \sum_{s'} P(s|s',a)V(s',a)$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end for
end loop // $\Delta > \theta$, a small positive number
/* Policy improvement */
policy_flag $\leftarrow$ true
for each $s \in S$ **do**
  last_action $\leftarrow \pi(s)$
  $\pi(s) \leftarrow \arg\min_a \left[r + \gamma \sum_{s'} P(s|s',a)V(s',a)\right]$
  if last_action $\neq \pi(s)$ **then**
    policy_flag $\leftarrow$ false
  end if
  if policy_flag == true **then**
    return $V \leftarrow v_*, \pi \leftarrow \pi_*$
  else
    go to line 4
  end if
end for

3.3.3 Analysis of time complexity

In Algorithm 2, the convergence of PPDO relates to value function $V(s)$. During the policy evaluation process, the computation of $V(s)$ in each state requires the system transfer probability and value functions be mutually independent. If the algorithm satisfies $\Delta<\theta$ within a smaller range of constants, the optimal time complexity of this process is $O(n^2)$. For policy improvement, it is necessary to select the action and compute $V(s)$. Generally, the time complexity of the process is $O(n^3)$. Overall, the time complexity of Algorithm 2 is $O(n^3)$.

3.4 Improvement of PPDO

In the MDP model established above, because the CEC system must engage in the collaborative computation of local devices and ESs, we implement privacy tasks to guarantee the protection of privacy. This leads to exploration with a high temporal overhead during policy iteration.

To address this issue, we accelerate the policy iteration by reducing action set A. Specifically, given a sufficiently small number of tasks, the tasks of the user devices can be completely offloaded to the ES for processing, and the selection of local devices in A can be reduced. Similarly, when the CPU processing capability of the ES is sufficiently large, many tasks are offloaded to the ES for processing. Thus, the selection of local devices in A decreases. In addition, when the CPU processing capability of the local device increases, the local device and ES adopt collaborative computation to execute tasks, while some farther distant ESs are not selected by the user. In this case, the ESs farther from this user can be removed from A. As the size of A decreases, the time complexity of the algorithm is reduced to $O(n^2)$.

The optimization component of PPDO is detailed in Algorithm 3. In lines 5 and 6, if the number of tasks is sufficiently small or the CPU processing capability of the ES is sufficiently large, the action a that executes the task locally is removed from A. Likewise, in lines 8 and 9, when the CPU processing capability of the local device is sufficiently large, actions are removed from A as the corresponding tasks are offloaded. Simultaneously, the reduced A^* is updated to action set A. According to Eq. (12), if the risk of usage pattern privacy leakage satisfies $l_{\mathrm{u}} = 0$, privacy tasks associated with usage pattern privacy can be omitted. Therefore, Algorithm 3 can be implemented within Algorithm 2 to ensure the efficiency of selection operations.

Algorithm 3 Improved policy iteration algorithm

1: **Input:** action set A, task number u_k, edge CPU capability f^{ES}, and local CPU capability f^{local}
2: **Output:** updated action set A
3: **loop**
4: **for** all $a \in A$ **do**
5: **if** $u_k \leq u_{\mathrm{thr}}$ or $f^{\mathrm{ES}} \geq f^{\mathrm{ES}}_{\mathrm{thr}}$ **then**
6: Remove action a for local execution of tasks
7: **end if**
8: **if** $f^{\mathrm{local}} \geq f^{\mathrm{local}}_{\mathrm{thr}}$ **then**
9: Remove action a for $d_{ij} > d_{\mathrm{thr}}$
10: **end if**
11: **end for**
12: **return** $A \leftarrow A^*$
13: **end loop**

4 Performance evaluation

4.1 Simulation setup and data preparation

The simulation was implemented in Java (based on JDK1.8) and the IntelliJ IDEA software (version 2019.3.3). The number of user devices (i.e., local devices) was alternated between 50 and 100. In addition, 30 edge servers were used in the simulation. To verify the effectiveness of PPDO, we considered the total delay metric, which encompasses the delay associated with both execution and privacy protection.

The parameters used in the simulations are listed in Table 2, where f^{local} is the CPU processing capability of the MU, ranging from 0.5 to 2.5 GHz, and s^{local} is the storage capacity of the MU, set to 1000 KB. The processing capability f^{ES} and storage capacity s^{ES} of the ES were set to (2.5, 4.5) GHz and (5000, 10 000) KB, respectively. s_k is the data size of task k, set to (10, 100) KB.

Table 2 Parameter setting

Parameter	Value
f^{local}	(0.5, 2.5) GHz
s^{local}	1000 KB
f^{ES}	(2.5, 4.5) GHz
s^{ES}	(5000, 10 000) KB
s_k	(10, 100) KB
δ_k	0.8*
c_k	$s_k\delta_k$
s^{privacy}	(5, 50) KB
δ^{privacy}	0.6*
c^{privacy}	$s^{\text{privacy}}\delta^{\text{privacy}}$
B_{w}	500 MHz
p	200 mW
σ	–100 mW
A_{s}	2.5
B	0.5
θ	1.6
a_{l}	0
b_{l}	2
h	0.5

* Number of cycles per unit data size

c_k is the number of CPU cycles for task k, taking the value of $s_k\delta_k$, where δ_k is the number of cycles per data unit for a normal task k and $\delta_k = 0.8$. For the privacy task, the data size s^{privacy} fell within (5, 50) KB and the number of CPU cycles was computed as $c^{\text{privacy}} = s^{\text{privacy}}\delta^{\text{privacy}}$. To guarantee the effectiveness of a privacy task, we tuned δ^{privacy} to 0.6, which was smaller than δ_k, to avoid augmenting the relative importance of privacy tasks. The network parameters are B_{w}, p, σ, A_{s}, and B.

Moreover, θ is a small positive number in Algorithm 2, and a_{l}, b_{l}, and h are constants in Eq. (11). The location information of local devices and ESs was selected from the EUA dataset (Lai et al., 2018), containing the location data of mobile users and edge servers in Melbourne City CBD.

4.2 Comparative algorithms

The proposed PPDO algorithm was compared with the following algorithms:

1. Random offloading algorithm. Without considering location distance, the algorithm randomly generates offloading decisions for user devices and then traverses all ES nodes to assign the offloading or local execution of tasks. The time complexity of this algorithm is $O(mn)$.

2. Genetic algorithm. The genetic algorithm uses a chromosome structure to represent the offloading strategy for tasks in MEC. The algorithm generates chromosomes and subsequently initiates a crossover mutation process. We set the chromosome size to 200, the crossover probability to 0.7, and the mutation probability to 0.2 (Wang ZB et al., 2023). The time overhead relates to the population size. In general, the time complexity is bounded by $O(n^2)$.

3. Policy-iteration-based algorithm. This algorithm solves the MDP problem using policy iteration that generates offloading decisions for the user. It does not account for collaborative computing between the ES and local device in terms of a full offloading of tasks to the ES (Wang WX et al., 2020).

Note that the latter two algorithms were combined with the proposed privacy-preserving scheme for comparison.

4.3 Comparison results

To verify the effectiveness of the proposed algorithm, we initially conducted a comparison in terms of the task number, edge CPU, and local device CPU to minimize the total delay. Next, we investigated the effect of the distance threshold on the risk of privacy leakage. Subsequently, a comparative simulation of the impact of privacy protection was conducted. Finally, the impact of the discount factor on algorithm performance was examined via simulation.

4.3.1 Evaluation of total delay

We compared the total delay incurred by each algorithm after implementing our privacy protection scheme. Three groups of comparison simulations were set up in terms of the number of tasks, CPU processing capability of the ES, and CPU processing capability of the local device. Fig. 5 illustrates the variation of total delay in the CEC under different conditions.

Fig. 5a shows that as the number of tasks generated by the mobile device increased, the total delay grew in all four algorithms. As random offloading cannot be rationalized in a collaborative computing system, the total delay of the GA algorithm was less than that of the randomized algorithm and greater than that of the MDP-based algorithm. Furthermore, the total delay under the PPDO algorithm became lower than that under Wang WX et al. (2020)'s method as the number of tasks increased. This can be attributed to the ability of PPDO to use local devices for collaborative computation.

As shown in Fig. 5b, following 30 tasks, the total delay incurred by the four algorithms decreased as the CPU processing capability of the ES increased. For higher processing capabilities, the policy-iteration-based algorithm (Wang WX et al., 2020) and PPDO exhibited similar performance. PPDO offloaded most tasks to the ES for execution when the CPU processing capability of the ES was sufficiently high. The genetic algorithm used collaborative computation between the local device and ES when the CPU of the ES was sufficiently small, incurring a total delay lower than that of Wang WX et al. (2020)'s method. Owing to its random characteristics, the random offloading algorithm obtained the worst results.

Fig. 5c illustrates the results for 30 tasks with respect to the CPU processing power of the local device. The total delay exhibited a decreasing trend for the random offloading, genetic, and PPDO algorithms. In contrast, the algorithm of Wang WX et al. (2020) could not take advantage of the local CPU processing capabilities, as it offloaded all tasks to the ES. When the chromosome of the genetic algorithm was set to zero, the tasks were executed on the local device (Wang ZB et al., 2023). Similarly, the PPDO algorithm may allow users to execute tasks at the local device to realize collaborative computation. Consequently, the total delay incurred by PPDO was less than that incurred by the genetic algorithm.

Overall, the MDP-based policy iteration algorithm outperformed all other algorithms with respect to the number of tasks, the CPU processing capability of the ES, and the processing capability of the local device. The PPDO algorithm achieved optimal performance by taking advantage of collaborative computation, thereby minimizing the total delay.

4.3.2 Impact on risk of privacy leakage

The privacy protection scheme proposed in this study offloaded privacy tasks to protect user privacy, with the number of privacy tasks being determined by the risk of privacy leakage. According to Section 2.1.3, the distance threshold d_{thr}, which affects the risk of privacy leakage, can be set to the average distance d_{avg} between the mobile device and all ESs, the smallest distance d_{min}, or the largest distance d_{max}. Fig. 6 depicts the risks of privacy leakage calculated according to Eqs. (11) and (12) under different values of d_{thr}.

Fig. 6a shows the leakage risk of usage pattern privacy l_{u} for different d_{thr}'s. As the number of users increased, l_{u} was observed to increase for both $d_{\mathrm{thr}}=d_{\mathrm{min}}$ and $d_{\mathrm{thr}}=d_{\mathrm{avg}}$. According to Eq. (12), l_{u} was always 0 at $d_{\mathrm{thr}}=d_{\mathrm{max}}$. Fig. 6b shows the risk of location privacy leakage l_{l} for different d_{thr}'s. For $d_{\mathrm{thr}}=d_{\mathrm{avg}}$ and $d_{\mathrm{thr}}=d_{\mathrm{max}}$, l_{l} increased with the increase of the number of users, and l_{l} reached its maximum (i.e., 100%) for all users at $d_{\mathrm{thr}}=d_{\mathrm{max}}$; for d_{min}, most of the risk values were close to 0. It is because the risk of location privacy existed only for individual users with d_{ij}=d_{min}. Fig. 6c illustrates the total risk of privacy leakage. When $d_{\mathrm{thr}}=d_{\mathrm{max}}$, all users had only location privacy leakage risk, while when $d_{\mathrm{thr}}=d_{\mathrm{min}}$, the leakage risk was mostly location privacy.

In general, the value of d_{thr} affects the risk of privacy leakage for both location and usage pattern. To simultaneously investigate the leakage risks for both types of privacy, we adopted $d_{\mathrm{thr}}=d_{\mathrm{avg}}$ for the simulations.

4.3.3 Impact of privacy preservation

To examine the impact of privacy protection on the total delay, we considered a scenario in which

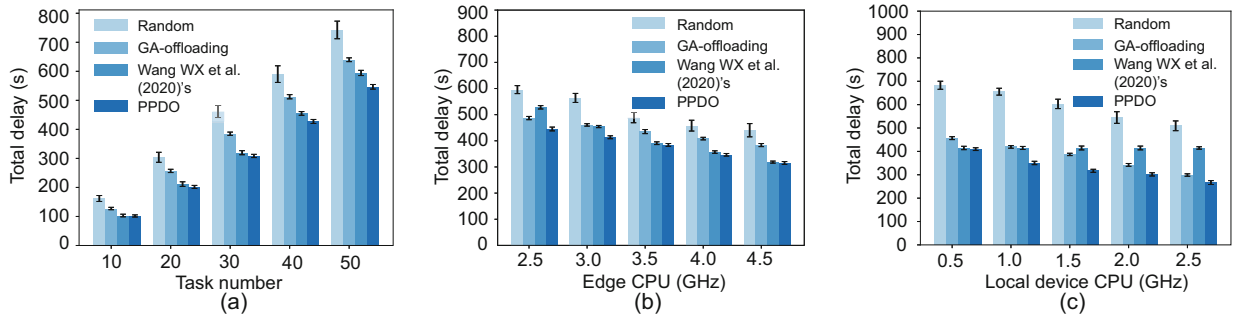

Fig. 5 Impact of the number of tasks (a), CPU of the ES (b), and CPU of the local device (c) on the total delay

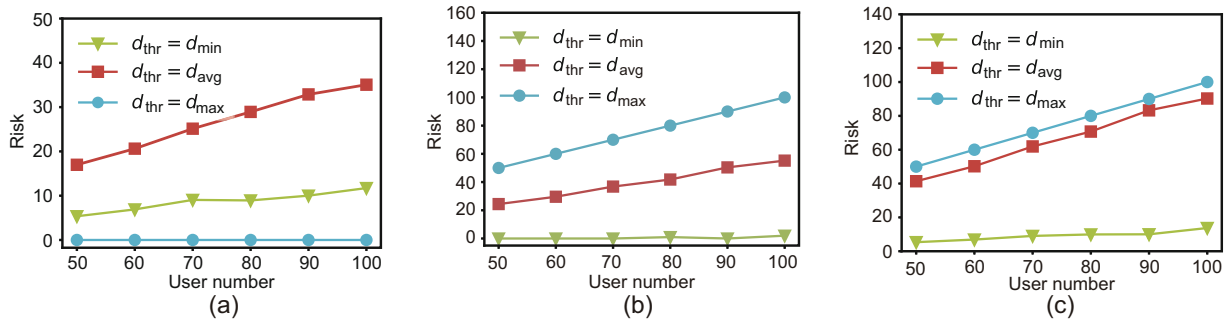

Fig. 6 Risks of privacy leakage under different d_{thr}'s: (a) risk of usage pattern privacy leakage; (b) risk of location privacy leakage; (c) total risk of privacy leakage

the protection of each type of privacy alone under different numbers of tasks was considered. To avoid the impact of device parameters on results, we set the CPU capabilities of the local device and ES at maxima of 2.5 and 4.5 GHz, respectively.

Table 3 presents the total delay comparison of the three algorithms with location privacy protection, showing that the PPDO algorithm incurred the smallest total delay. Local devices considered only the risk of location privacy leakage l_l and added additional privacy task queues for mobile devices accordingly. Although the total delay incurred by each algorithm increased with the increase of the number of tasks, it remained less than that incurred when both privacy protection schemes were implemented, as fewer privacy tasks were generated.

Table 4 presents a delay comparison of the three algorithms solely under usage pattern privacy protection. Mobile devices took into account only their risk of usage pattern privacy leakage l_u and added privacy tasks to the user device task queue based on l_u. Similarly, although the total delay incurred by each algorithm increased with the increase of the number of tasks, it remained less than that incurred when both privacy protection schemes were implemented. The PPDO algorithm also obtained the least total delay when using only usage pattern privacy protection.

Table 3 Total delay with location privacy protection

Task number	Total delay (s)		
	GA	Wang WX et al. (2020)'s method	PPDO
10	125.41	101.39	98.81
20	249.88	205.99	200.59
30	377.88	329.15	304.24
40	505.84	435.21	421.99
50	625.86	545.97	520.81

GA: genetic algorithm; PPDO: privacy-preservation-aware delay optimization

From the above results, we observed that the two privacy protection schemes had impacts on the total delay with location privacy protection, resulting in greater total delay. However, the PPDO algorithm balanced the total delay irrespective of the privacy protection scheme.

Table 4 Total delay with usage pattern privacy protection

Task number	Total delay (s)		
	GA	Wang WX et al. (2020)'s method	PPDO
10	122.91	98.57	95.36
20	247.93	198.88	194.76
30	370.69	305.18	301.08
40	495.96	433.60	412.65
50	618.86	546.48	528.69

GA: genetic algorithm; PPDO: privacy-preservation-aware delay optimization

4.3.4 Impact of γ

Because the PPDO algorithm is primarily built on the MDP model, the discount factor γ in Eq. (22) is proportional to the impact of the current decision on the future state. Accordingly, we varied the setting of γ to evaluate its effect on algorithm performance. For this experiment, we set the task number to 30, CPU capability of the local device to 2.5 GHz, and CPU capability of the ES to 4.5 GHz.

As shown in Fig. 7, as γ grew from 0.1 to 0.9, the total delay of the PPDO algorithm was minimized at $\gamma = 0.9$. This indicated that PPDO relies more on the future state than the current state when computing $V(s, a)$. In our simulation, the PPDO algorithm yielded optimal performance at $\gamma = 0.9$.

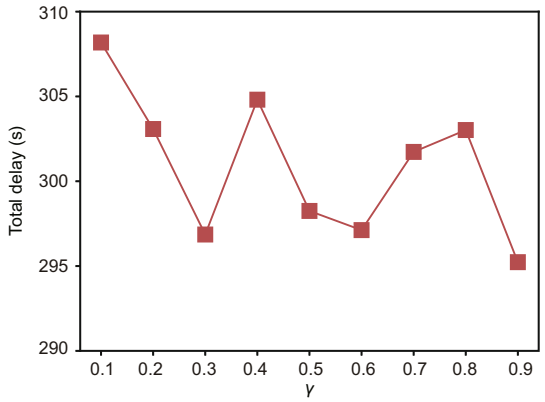

Fig. 7 Impact of γ on the total delay using the PPDO algorithm

4.4 Discussion of improvement of PPDO

We previously mentioned that the proposed policy iteration algorithm may incur a high time overhead. To investigate the performance improvement in PPDO, we extended our simulation to time overhead in terms of running time. We set the user device number to 50, and set the task number to 10, CPU processing capability of the ES to 4.5 GHz, and local device CPU to 2.5 GHz, separately.

Fig. 8 presents the running time obtained by the two algorithms. Owing to the reduction in action set A, the traversal of local devices and ESs was reduced during policy evaluation, thereby accelerating the iterative process. Under these conditions, there was a significant decrease in running time.

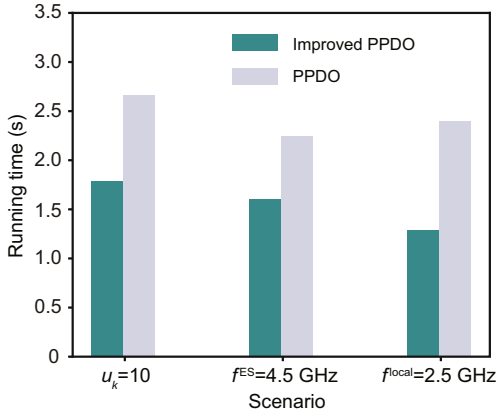

Fig. 8 Comparison results in terms of running time

However, because the improved PPDO algorithm no longer included the action of the local device, it became equivalent to that in Wang WX et al. (2020), which offloaded all tasks to the ES when the task number was sufficiently small (10) or the capability of the ES was sufficiently large (4.5 GHz). Furthermore, if the local device CPU was maximized at 2.5 GHz, the ESs that satisfied the $d_{ij} > d_{\text{thr}}$ distance requirement were removed from A, leaving fewer available ESs. Because the risk of privacy leakage of the user's usage pattern privacy $l_{\text{u}} = 0$ was calculated according to Eq. (12), the number of privacy tasks was reduced, resulting in a change in the total delay.

Fig. 9 presents a comparison of the total delay between the improved PPDO and PPDO. Under certain conditions, when the number of tasks was small and the device's CPU processing capability was high, the improved PPDO required more running time compared to PPDO, and the total delay was slightly larger than that of PPDO.

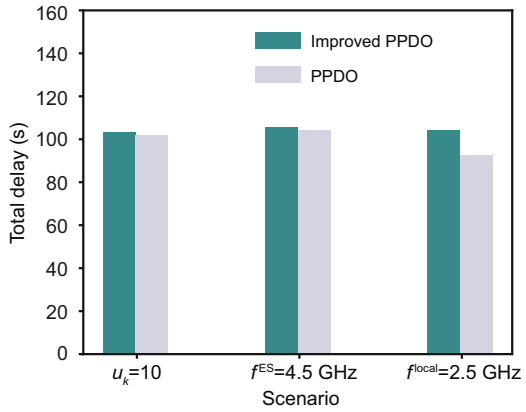

Fig. 9 Comparison of the total delay between improved PPDO and PPDO

5 Conclusions

This study primarily investigated the balance between privacy protection and delay in CEC systems. In contrast to prior studies, we investigated the problem of privacy preservation with the objective of minimizing the total delay in task offloading. The preservation scheme calculated the privacy leakage risk and offloaded privacy tasks accordingly. Subsequently, the issue of minimizing the delay incurred by task offloading was formulated as an MDP problem. We developed a policy-iteration-based privacy-preservation-aware offloading algorithm to reduce the total latency of task offloading while protecting user privacy. Furthermore, we extended PPDO by decreasing the exploration time in the action set. The results showed that the proposed algorithm minimized the delays without compromising privacy protection while offloading the tasks. The complexity of the improved PPDO was compared and discussed to optimize performance in terms of running time.

In future work, we will consider the use of a changeable network channel model in CEC. Furthermore, to fit into the real-time scenario, it is important to integrate the concept of adaptiveness into the current algorithm. This can be achieved by incorporating reinforcement learning to find a trade-off between privacy protection and delay.

Contributors

Chao JING guided the research. Jianwu XU designed the research and drafted the paper. Chao JING revised and finalized the paper.

Conflict of interest

Both authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are not openly available as they are part of an ongoing study.

References

Abbas N, Zhang Y, Taherkordi A, et al., 2018. Mobile edge computing: a survey. *IEEE Int Things J*, 5(1):450-465. https://doi.org/10.1109/JIOT.2017.2750180

Bai Y, Chen LX, Song LQ, et al., 2020. Risk-aware edge computation offloading using Bayesian Stackelberg game. *IEEE Trans Netw Serv Manag*, 17(2):1000-1012. https://doi.org/10.1109/TNSM.2020.2985080

Cao T, Qian ZZ, Wu K, et al., 2021. Service placement and bandwidth allocation for MEC-enabled mobile cloud gaming. Proc 22nd Int Symp on a World of Wireless, Mobile and Multimedia Networks, p.179-188. https://doi.org/10.1109/WoWMoM51794.2021.00031

Chen SG, Zheng YM, Lu WF, et al., 2020. Energy-optimal dynamic computation offloading for Industrial IoT in fog computing. *IEEE Trans Green Commun Netw*, 4(2):566-576. https://doi.org/10.1109/TGCN.2019.2960767

Chu WB, Jia XM, Yu ZW, et al., 2023. Joint service caching, resource allocation and task offloading for MEC-based networks: a multi-layer optimization approach. *IEEE Trans Mob Comput*, 23(4):2958-2975. https://doi.org/10.1109/TMC.2023.3268048

Dong LB, Gao HH, Wu WL, et al., 2023. Dependence-aware edge intelligent function offloading for 6G-based IoV. *IEEE Trans Intell Transp Syst*, 24(2):2265-2274. https://doi.org/10.1109/TITS.2022.3148229

Gao HH, Wang XJ, Wei W, et al., 2023a. Com-DDPG: task offloading based on multiagent reinforcement learning for information-communication-enhanced mobile edge computing in the Internet of Vehicles. *IEEE Trans Veh Technol*, 73(1):348-361. https://doi.org/10.1109/TVT.2023.3309321

Gao HH, Huang WQ, Liu T, et al., 2023b. PPO2: location privacy-oriented task offloading to edge computing using reinforcement learning for intelligent autonomous transport systems. *IEEE Trans Intell Transp Syst*, 24(7):7599-7612. https://doi.org/10.1109/TITS.2022.3169421

He T, Ciftcioglu EN, Wang SQ, et al., 2017. Location privacy in mobile edge clouds. Proc IEEE 37th Int Conf on Distributed Computing Systems, p.2264-2269. https://doi.org/10.1109/ICDCS.2017.39

He XF, Liu J, Jin RC, et al., 2017. Privacy-aware offloading in mobile-edge computing. IEEE Global Communications Conf, p.1-6. https://doi.org/10.1109/GLOCOM.2017.8253985

He XF, Jin RC, Dai HY, 2020. Peace: privacy-preserving and cost-efficient task offloading for mobile-edge computing. *IEEE Trans Wirel Commun*, 19(3):1814-1824. https://doi.org/10.1109/TWC.2019.2958091

Hua W, Zhou ZY, Huang LY, 2023. Location privacy-aware offloading for MEC-enabled IoT: optimality and heuristics. *IEEE Int Things J*, 10(21):19270-19281. https://doi.org/10.1109/JIOT.2023.3281609

Ksentini A, Taleb T, Chen M, 2014. A Markov decision process-based service migration procedure for follow me cloud. IEEE Int Conf on Communications, p.1350-1354. https://doi.org/10.1109/ICC.2014.6883509

Lai P, He Q, Abdelrazek M, et al., 2018. Optimal edge user allocation in edge computing with variable sized vector bin packing. Proc 16th Int Conf on Service-Oriented Computing, p.230-245. https://doi.org/10.1007/978-3-030-03596-9_15

Lee S, Lee S, Lee SS, 2021. Deadline-aware task scheduling for IoT applications in collaborative edge computing. *IEEE Wirel Commun Lett*, 10(10):2175-2179. https://doi.org/10.1109/LWC.2021.3095496

Lin P, Song QY, Wang D, et al., 2021. Resource management for pervasive-edge-computing-assisted wireless VR streaming in Industrial Internet of Things. *IEEE Trans Ind Inform*, 17(11):7607-7617. https://doi.org/10.1109/TII.2021.3061579

Mach P, Becvar Z, 2017. Mobile edge computing: a survey on architecture and computation offloading. *IEEE Commun Surv Tut*, 19(3):1628-1656. https://doi.org/10.1109/COMST.2017.2682318

Mao S, Liu L, Zhang N, et al., 2022. Reconfigurable intelligent surface-assisted secure mobile edge computing networks. *IEEE Trans Veh Technol*, 71(6):6647-6660. https://doi.org/10.1109/TVT.2022.3162044

Mao YY, Zhang J, Song SH, et al., 2017. Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. *IEEE Trans Wirel Commun*, 16(9):5994-6009. https://doi.org/10.1109/TWC.2017.2717986

Min MH, Wan XY, Xiao L, et al., 2019. Learning-based privacy-aware offloading for healthcare IoT with energy harvesting. *IEEE Int Things J*, 6(3):4307-4316. https://doi.org/10.1109/JIOT.2018.2875926

Ouyang T, Li R, Chen X, et al., 2019. Adaptive user-managed service placement for mobile edge computing: an online learning approach. IEEE Conf on Computer Communications, p.1468-1476. https://doi.org/10.1109/INFOCOM.2019.8737560

Qian LP, Wu WC, Lu WD, et al., 2021. Secrecy-based energy-efficient mobile edge computing via cooperative non-orthogonal multiple access transmission. *IEEE Trans Commun*, 69(7):4659-4677. https://doi.org/10.1109/TCOMM.2021.3070620

Qin XD, Li B, Ying L, 2021. Distributed threshold-based offloading for large-scale mobile cloud computing. IEEE Conf on Computer Communications, p.1-10. https://doi.org/10.1109/INFOCOM42981.2021.9488821

Ren JK, Yu GD, He YH, et al., 2019. Collaborative cloud and edge computing for latency minimization. *IEEE Trans Veh Technol*, 68(5):5031-5044. https://doi.org/10.1109/TVT.2019.2904244

Sahni Y, Cao JN, Yang L, 2019. Data-aware task allocation for achieving low latency in collaborative edge computing. *IEEE Int Things J*, 6(2):3512-3524. https://doi.org/10.1109/JIOT.2018.2886757

Taleb T, Dutta S, Ksentini A, et al., 2017. Mobile edge computing potential in making cities smarter. *IEEE Commun Mag*, 55(3):38-43. https://doi.org/10.1109/MCOM.2017.1600249CM

Wang F, Xu J, Wang X, et al., 2018. Joint offloading and computing optimization in wireless powered mobile-edge computing systems. *IEEE Trans Wirel Commun*, 17(3):1784-1797. https://doi.org/10.1109/TWC.2017.2785305

Wang JD, Zhao L, Liu JJ, et al., 2021. Smart resource allocation for mobile edge computing: a deep reinforcement learning approach. *IEEE Trans Emerg Top Comput*, 9(3):1529-1541. https://doi.org/10.1109/TETC.2019.2902661

Wang TS, Li Y, Wu Y, 2021. Energy-efficient UAV assisted secure relay transmission via cooperative computation offloading. *IEEE Trans Green Commun Netw*, 5(4):1669-1683. https://doi.org/10.1109/TGCN.2021.3099523

Wang WX, Ge SX, Zhou XB, 2020. Location-privacy-aware service migration in mobile edge computing. IEEE Wireless Communications and Networking Conf, p.1-6. https://doi.org/10.1109/WCNC45663.2020.9120551

Wang ZB, Pang XY, Chen YH, et al., 2019. Privacy-preserving crowd-sourced statistical data publishing with an untrusted server. *IEEE Trans Mob Comput*, 18(6):1356-1367. https://doi.org/10.1109/TMC.2018.2861765

Wang ZB, Sun YN, Liu DF, et al., 2023. Location privacy-aware task offloading in mobile edge computing. *IEEE Trans Mob Comput*, 23(3):2269-2283. https://doi.org/10.1109/TMC.2023.3254553

Yang XM, Luo H, Sun Y, et al., 2020. Energy-efficient collaborative offloading for multiplayer games with cache-aided MEC. IEEE Int Conf on Communications, p.1-7. https://doi.org/10.1109/ICC40277.2020.9148751

Yousaf K, Mehmood Z, Saba T, et al., 2018. A novel technique for speech recognition and visualization based mobile application to support two-way communication between deaf-mute and normal peoples. *Wirel Commun Mob Comput*, 2018:1013234. https://doi.org/10.1155/2018/1013234

Zhao P, Jiang HB, Lui JCS, et al., 2018. P^3-LOC: a privacy-preserving paradigm-driven framework for indoor localization. *IEEE/ACM Trans Netw*, 26(6):2856-2869. https://doi.org/10.1109/TNET.2018.2879967

Zhao P, Tao JW, Lui K, et al., 2023. Deep reinforcement learning-based joint optimization of delay and privacy in multiple-user MEC systems. *IEEE Trans Cloud Comput*, 11(2):1487-1499. https://doi.org/10.1109/TCC.2022.3140231