A Potential Game Theoretic Approach to Computation Offloading Strategy Optimization in End-Edge-Cloud Computing

Yan Ding, *Student Member, IEEE*, Kenli Li, *Senior Member, IEEE*, Chubo Liu, *Member, IEEE*, and Keqin Li, *Fellow, IEEE*

Abstract—Integrating user ends (UEs), edge servers (ESs), and the cloud into end-edge-cloud computing (EECC) can enhance the utilization of resources and improve quality of experience (QoE). However, the performance of EECC is significantly affected by its architecture. In this article, we classify EECC into two computing architectures types according to the visibility and accessibility of the cloud to UEs, i.e., hierarchical end-edge-cloud computing (Hi-EECC) and horizontal end-edge-cloud computing (Ho-EECC). In Hi-EECC, UEs can offload their tasks only to ESs. When the resources of ESs are exhausted, the ESs request the cloud to provide resources to UEs. In Ho-EECC, UEs can offload their tasks directly to ESs and the cloud. In this article, we construct a potential game for the EECC environment, in which each UE selfishly minimizes its payoff, study the computation offloading strategy optimization problems, and develop two potential game-based algorithms in Hi-EECC and Ho-EECC. Extensive experiments with real-world data are conducted to demonstrate the performance of the proposed algorithms. Moreover, the scalability and applicability of the two computing architectures are comprehensively analyzed. The conclusions of our work can provide useful suggestions for choosing specific computing architectures under different application environments to improve the performance of EECC and QoE.

Index Terms—Computation offloading, end-edge-cloud computing (EECC), hierarchical EECC, horizontal EECC, potential game

1 Introduction

1.1 Motivation

The vigorous development of Internet of Everything (IoE) and artificial intelligence technologies has given rise to an intelligence era for human society [1]. For example, Hikvision has developed surveillance cameras that no longer only obtain videos and images as in the past, but have the ability to recognize and track objects. Mobile phones (e.g., Huawei, Apple, and Xiaomi) have also evolved from traditional communication devices to important carriers running various applications, such as electronic payment, home management, virtual reality, and other intelligent applications. Intelligent applications require the support of powerful computing. However, the resources possessed by user ends (UEs), such as mobile phones and Internet of Things (IoT) devices cannot run intelligent applications efficiently.

Deploying computing resources near the network edge is considered a promising solution to the above issue. Edge computing (EC) cannot only provide low-latency services for UEs but also guarantee the data security of UEs [2]. EC has been widely studied in many directions, such as computation offloading [3], caching [4], resource allocation [5], [6], and privacy protection [7]. However, the above work ignores an important fact, that is, edge servers (ESs) do not have the same ability to handle computation-intensive tasks as the cloud [8].

Although cloud computing (CC) has sufficient resources to support the requirements of computation-intensive tasks [9], it cannot solve the issue of long delay caused by data transmission [10]. Due to the shorter transmission distance and higher transmission rate between UEs and ESs, EC can reduce data transmission delay and is suitable for providing services for handling latency-sensitive tasks. Therefore, cooperation between EC and CC can better meet various user demands. Edge-cloud computing has been studied in various work [11], [12], [13], [14]. However, when the network is unstable or the resource competition between UEs is tight, it is better for a UE to rely on its own ability to handle some tasks. Therefore, UEs, ESs, and the cloud can be integrated into end-edge-cloud computing (EECC), which cannot only enhance the utilization of resources but also improve quality of experience (QoE) while ensuring quality of service (QoS).

- *Yan Ding, Kenli Li, and Chubo Liu are with the College of Information Science and Engineering, Hunan University, Changsha, Hunan 410082, China, and also with the National Supercomputing Center in Changsha, Changsha, Hunan 410082, China. E-mail: {ding, lkl, liuchubo}@hnu.edu.cn.*
- *Keqin Li is with the College of Information Science and Engineering, Hunan University, and the National Supercomputing Center in Changsha, Changsha, Hunan 410082, China, and also with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA. E-mail: lik@newpaltz.edu.*

Manuscript received 6 Feb. 2021; revised 1 July 2021; accepted 10 Sept. 2021. Date of publication 14 Sept. 2021; date of current version 28 Oct. 2021.
This work was partially supported by the National Outstanding Youth Science Program of National Natural Science Foundation of China under Grant 61625202, the National Key Research and Development Program of China under Grant 2018YFB1701403, the Programs of National Natural Science Foundation of China under Grants 61876061, 62072165, U19A2058, and 61702170, and the Postgraduate Scientific Research Innovation Project of Hunan Province under Grant CX20200435. This work was also sponsored by the Open Research Projects of Zhejiang Lab under Grant 2020KE0AB01.
(Corresponding authors: Kenli Li and Keqin Li.)
Recommended for acceptance by R. M. Badia.
Digital Object Identifier no. 10.1109/TPDS.2021.3112604

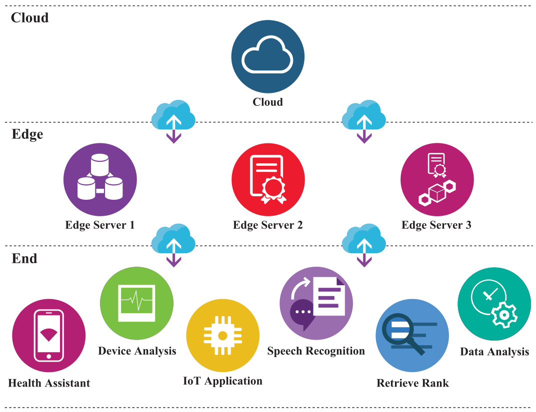


Fig. 1. Illustration of Hi-EECC.

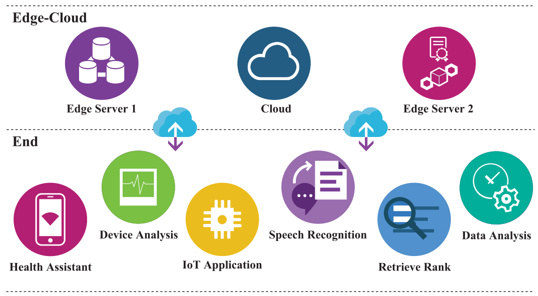


Fig. 2. Illustration of Ho-EECC.

Although some work has verified the effectiveness of EECC [15], [16], [17], its performance is significantly affected by its architecture, which has not been studied. Motivated by the above reality, we classify EECC into the following two types of architectures according to the visibility and accessibility of the cloud to UEs:

- Hierarchical end-edge-cloud computing (Hi-EECC). As shown in Fig. 1, Hi-EECC is a three-tier architecture, and the cloud is invisible to UEs. In Hi-EECC, UEs can offload their tasks only to ESs. When the resources of ESs are exhausted or the QoS demands of UEs cannot be satisfied by the ESs, the tasks are uploaded to the cloud by the ESs. The service provided by the cloud is transparent for UEs.
- Horizontal end-edge-cloud computing (Ho-EECC). As shown in Fig. 2, Ho-EECC is a two-tier architecture. Since the cloud resources are visible and accessible to UEs, both the cloud and ESs are in the second layer of Fig. 2, i.e., Edge-Cloud layer. In Ho-EECC, the cloud does not need to rely on ESs to provide services to UEs. UEs can request ESs and the cloud directly according to their own preferences.

In this paper, we investigate the computation offloading strategy optimization problems in Hi-EECC and Ho-EECC, and analyze the impact of the architectures on UEs and ESs. We comprehensively study the scalability and applicability of the two computing architectures in terms of the energy consumption of UEs, time consumption of UEs, resource utilization rate of ESs, application type, and user scale.

1.2 Our Contributions

To the best of our knowledge, this paper is the first work to optimize computation offloading strategy for UEs, and investigate the performance of EECC in different computing architectures. The contributions are as follows.

- The computation offloading strategy optimization problems in Hi-EECC and Ho-EECC are investigated, and the impact of the two computing architectures on UEs and ESs is analyzed in detail.
- Considering the selfishness of UEs, we construct a potential game for the EECC environment, in which each UE selfishly minimizes its payoff. We also develop two potential game-based algorithms according to the characteristics of computing architectures. The existence of Nash equilibrium, the convergence of the algorithms, and the performance of the algorithms are theoretically analyzed in detail.
- Extensive experiments with real-world data are conducted to demonstrate the performance of the proposed algorithms. The scalability and applicability of two computing architectures are comprehensively analyzed in detail. Three important conclusions are presented for choosing specific computing architectures in the different application scenario.

The remaining content is outlined as follows. In Section 2, the related work is reviewed. System models are detailed in Section 3. The EECC game is formulated in Section 4. The potential game-based computation offloading algorithms are described in Section 5. The convergence and performance of the algorithms are theoretically demonstrated in Section 6. We conduct extensive experiments using real-word data to verify the proposed algorithms and theorems, and to analyze the two computing architectures in Section 7. Section 8 provides the conclusions of this paper and our future work.

2 Related Work

The computation offloading strategy optimization problem in EC has consistently been a hot research topic in industry and academia, and has been investigated extensively. Wu *et al.* [18] studied the problem in a multi-channel wireless interference environment, and proposed a distributed algorithm to minimize the total delay of all UEs. You *et al.* [19] optimized the offloading strategy by minimizing the weighted sum of energy consumption of UEs under the constraint of computation delay, and they considered the time-division multi-access and orthogonal frequency-division multi-access communication modes. Chen *et al.* [20] improved the long-term performance of EC by using the Lyapunov optimization technique, and proposed an online algorithm without requiring future information about user demands. Hu *et al.* [21] proposed a greedy-based pruning algorithm to select UEs that should offload their tasks to ESs and developed a non-cooperative game-based iteration algorithm to determine the final strategy.

However, the computing capacity of EC is limited relative to the cloud. It is necessary to integrate EC and CC into a collaborative computing architecture, thus improving the performance of the architecture and introducing higher levels of flexibility for various demands of UEs. Some work has studied the computation offloading optimization problem in the edge-cloud computing architecture. For example, Ren *et al.* [11] investigated the computation offloading optimization problem in the hierarchical edge-cloud computing architecture and developed a convex-based algorithm to decide the task slipping strategy. Shah-Mansouri *et al.* [12] developed a potential game-based algorithm to optimize the strategy in the horizontal edge-cloud computing architecture. Du *et al.* [13] considered the communication cost between co-resident and non-co-resident tasks, and designed an algorithm to obtain a suboptimal strategy. Fantacci *et al.* [14] formulated the problem as a queueing system model and determined the strategy by maximizing the rate of UEs whose QoS can be satisfied.

It is also necessary for UEs to perform some computations locally, which copes with wireless network problems, such as network disconnection and instability. Moreover, UEs can process real-time tasks (such as emergency stop and failure recovery) that are very sensitive to delay [22]. Very little work has studied the computation offloading strategy optimization problem in EECC. For example, Hong *et al.* [22] studied the multi-hop computation offloading strategy optimization problem. Peng *et al.* [23] optimized the offloading strategy based on the strength Pareto evolutionary algorithm. Sun *et al.* [24] developed a hierarchical heuristic approach to make offloading decisions. Wang *et al.* [15] investigated the application of EECC to an underwater acoustic sensor network.

The effectiveness of EECC in improving the overall performance of the computing architecture has been widely demonstrated [15], [16], [17]. However, the performance of different EECC architectures has not been studied. Specifically, there is no work that investigates the computation offloading optimization problem under the different computing architectures, and summarizes how to choose a specific architecture of EECC for the different application scenario. To fill this research gap, this paper develops potential game-based algorithms for UE optimizing offloading strategies, provides a performance analysis of the two types of EECCs under the different application scenario, and analyzes the impact of various factors on the cost of UEs and the resource utilization of ESs. The main observations concluded from the experiments can provide some useful suggestions for improving the QoE of UEs with various requirements. Section 7 explains these interesting observations in detail.

3 Models

3.1 System Model

Fig. 3 depicts the scenario studied in this paper. Table 1 lists the parameters and their definitions in this paper. We assume that there is a group of UEs $\mathcal{N}$ that can be served by a group of ESs $\mathcal{M}$. We use UE$_n$ ($n \in [1, |\mathcal{N}|]$) and ES$_m$ ($m \in [1, |\mathcal{M}|]$) to represent the nth UE and mth ES, respectively. Additionally, there is a cloud that can provide computing resources to UEs. However, the visibility and accessibility of the cloud to the UE is different between Hi-EECC and Ho-EECC. In this paper, the demand of UE$_n$ not only reduces its cost but also requests the service delay to be less than the deadline determined by the UE. Because the

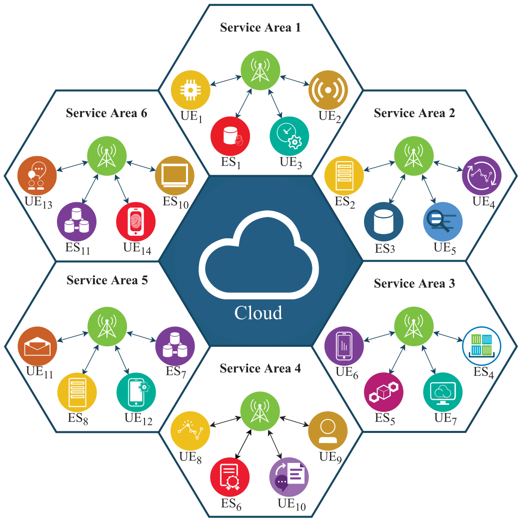


Fig. 3. Illustration of the scenario studied in this paper.

resources of ESs are limited, the demands of UEs may not be satisfied when the requirements exceed the resource capacity of ESs. As shown in Fig. 1, in Hi-EECC, UEs can offload their tasks only to ESs to reduce their cost. A task of UE$_n$ is offloaded to ES$_m$ when the cost of UE$_n$ being served by the ES is less than the local execution cost. However, if UE$_n$'s deadline cannot be guaranteed, the task will be further uploaded to the cloud by ES$_m$. In other words, the service provided by the cloud in Hi-EECC is transparent for UEs.

As shown in Fig. 2, in Ho-EECC, UEs can offload their tasks to ESs or the cloud according to their requirements. In addition, offloading decisions made by a UE should not only reduce its cost but also ensure that the service delay is less than its deadline.

By using high-speed wireless communication technology, continuous service can be provided for UEs with mobility. However, the distance between communication entities has become the main factor that affects the data transmission rate. To reflect this reality, we assume that there is a set of service areas $\mathcal{I}$ and use $i \in [1, |\mathcal{I}|]$ to represent the ith service area. Moreover, if UE$_n$ and ES$_m$ are in the same service area, the ES can respond to the request of UE$_n$. As shown in Fig. 3, there are six service areas in the studied scenario. Since UE$_1$, UE$_2$, and UE$_3$ are in the 1st service area, they can initiate requests only to ES$_1$ in Hi-EECC. However, UEs can initiate requests to the cloud in Ho-EECC.

In EECC, the heterogeneous characteristics of UEs and ESs have always been challenges for optimizing computation offloading strategies. In this paper, UE$_n$ is specified by f_n, γ_n, a_n, σ_n, and σ'_n, where $f_n, \gamma_n, a_n, \sigma_n$, and σ'_n are the computing capability of UE$_n$ (i.e., CPU frequency, which is quantified by the number of cycles per second), the resource allocation weight parameter of UE$_n$, and a task of UE$_n$, the energy consumption (Joule, J) per second for UE$_n$ processing a_n, and the energy consumption per second for UE$_n$ uploading a_n, respectively. Since 1 Watt = 1 J/s, σ_n, σ'_n are CPU execution and data transmission power of UE$_n$. The resources of an ES are allocated proportionally to UEs. We use $\gamma_n > 0$ to denote the proportion of resources that UE$_n$ can obtain from ES$_m$ among all UEs that send a request to the ES, which can be determined by the payment level of

TABLE 1
Summary of Notations and Definitions

Notations	Definition
System Model	
UE_n	the nth UE, $1 \leq n \leq \lvert\mathcal{N}\rvert$
ES_m	the mth ES, $1 \leq m \leq \lvert\mathcal{M}\rvert$
i	the serial number of service area, $1 \leq i \leq \lvert\mathcal{I}\rvert$
f_n	the computing capability of UE_n, which is quantified by cycles/s
γ_n	the resource allocation weight parameter of UE_n
a_n	$\triangleq (\delta_n, \omega_n, \zeta_n, d_n)$, a task of UE_n
σ_n	the energy consumption for UE_n processing a_n, which is measured by J/s
σ'_n	the energy consumption for UE_n uploading a_n, which is measured by J/s
δ_n	the data size of a_n, which is measured by the number of bits
ω_n	the number of CPU cycles needed to complete a_n
ζ_n	the location of UE_n when a_n is executed
d_n	the QoS requirement of a_n
$\tilde{f}_m$	the computing capability of ES_m, which is quantified by cycles/s
$\tilde{r}_m$	the data transmission rate of ES_m, which is measured by bits/s
$\tilde{\zeta}_m$	the location of ES_m
$\hat{f}$	the computing capability of the cloud, which is quantified by cycles/s
SN_i	$\triangleq \{\mathrm{UE}_n \mid \zeta_n = i\}$, the set of UEs in the ith service area
SM_i	$\triangleq \{\mathrm{ES}_m \mid \tilde{\zeta}_m = i\}$, the set of ESs in the ith service area
λ_n	an indicator variable indicating whether to execute a_n locally
$\tilde{\lambda}_{n,m}$	an indicator variable indicating whether to upload a_n to ES_m
$\hat{\lambda}_n$	an indicator variable indicating whether to upload a_n to the cloud
Cost Model	
t_n	the computation delay of a_n executed locally
e_n	$= \sigma_n t_n$, the energy consumption of a_n executed locally
$f_{n,m}$	the computing resource of ES_m allocated to UE_n
$\tilde{t}_{n,m}$	the computation delay of a_n executed by ES_m
$\hat{t}_n$	the computation delay of a_n executed by the cloud
$r_{n,m}$	the data transmission rate of ES_m allocated to UE_n
$\tilde{t}'_{n,m}$	the communication delay for UE_n uploading a_n to ES_m
$\tilde{e}_{n,m}$	$= \sigma'_n \tilde{t}'_{n,m}$, the energy consumption for UE_n uploading a_n to ES_m
$\hat{t}'_{n,hi}$	the communication delay for ES_m uploading a_n to the cloud in Hi-EECC
$\hat{t}'_{n,ho}$	the communication delay for UE_n uploading a_n to the cloud in Ho-EECC
$\hat{e}_n$	$= \sigma'_n \hat{t}'_{n,ho}$, the energy consumption for UE_n uploading a_n to the cloud in Ho-EECC
$T_{n,hi}$	the delay of UE_n in Hi-EECC
$E_{n,hi}$	the energy consumption of UE_n in Hi-EECC
$T_{n,ho}$	the delay of UE_n in Ho-EECC
$E_{n,ho}$	the energy consumption of UE_n in Ho-EECC
ϕ_n	the individual preference of UE_n between energy consumption and time consumption
$C_{n,l}$	$= \phi_n t_n + (1 - \phi_n) e_n$, the cost of a_n executed locally
$\tilde{C}_{n,m}$	$= \phi_n(\tilde{t}_{n,m} + \tilde{t}'_{n,m}) + (1 - \phi_n)\tilde{e}_{n,m}$, the cost of a_n executed by ES_m
$C_{n,hi}$	$= \phi_n T_{n,hi} + (1 - \phi_n) E_{n,hi}$, the cost of UE_n in Hi-EECC
$C_{n,ho}$	$= \phi_n T_{n,ho} + (1 - \phi_n) E_{n,ho}$, the cost of UE_n in Ho-EECC
$C_{m,c}(\Lambda)$	$= \sum_{n=1}^{\lvert\mathcal{N}\rvert} \hat{t}_n \big(1 - \sum_{m=1}^{\lvert\mathcal{N}\rvert} \tilde{\lambda}_{n,m} - \lambda_n\big)\kappa_m$, the cost of ES_m
κ_m	the correlation parameter between $\hat{t}_n$ and the cost of ES_m
Potential Game Theory	
$\mathbb{R}^{\lvert\mathcal{M}\rvert+1}$	an euclidean space
K_n	all possible offloading strategies of the nth player (i.e., UE_n)
$\mathcal{K}$	$= K_1 \times K_2 \times \ldots \times K_{\lvert\mathcal{N}\rvert}$, all possible offloading strategy sets of $\lvert\mathcal{N}\rvert$ UEs
$\boldsymbol{\lambda_n}$	$= (\lambda_n, \tilde{\lambda}_{n,1}, \ldots, \tilde{\lambda}_{n,\lvert\mathcal{M}\rvert})$, the nth player derives a strategy between itself and ESs
Λ	$= (\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, \ldots, \boldsymbol{\lambda_{\lvert\mathcal{N}\rvert}})^T$, the offloading strategy set of $\lvert\mathcal{N}\rvert$ UEs
Λ_{-n}	$= (\boldsymbol{\lambda_1}, \ldots, \boldsymbol{\lambda_{n-1}}, \boldsymbol{\lambda_{n+1}}, \ldots, \boldsymbol{\lambda_{\lvert\mathcal{N}\rvert}})^T$, the strategies of $\lvert\mathcal{N}\rvert - 1$ players except for UE_n
$C(\boldsymbol{\lambda_n}, \Lambda_{-n})$	the cost of UE_n adopting $\boldsymbol{\lambda_n}$ when Λ_{-n} is given in the EECC game
$\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n})$	a potential function of $\boldsymbol{\lambda_n}$ when Λ_{-n} is given
Λ^*	$= (\boldsymbol{\lambda_1^*}, \boldsymbol{\lambda_2^*}, \ldots, \boldsymbol{\lambda_{\lvert\mathcal{N}\rvert}^*})^T$, the Nash equilibrium of the EECC game
Λ_o	$= (\boldsymbol{\lambda_{o,1}}, \boldsymbol{\lambda_{o,2}}, \ldots, \boldsymbol{\lambda_{o,\lvert\mathcal{N}\rvert}})^T$, the best strategy set between local processing and ESs processing for all UEs
$\hat{\boldsymbol{\lambda}}$	$= (\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_{\lvert\mathcal{N}\rvert})^T$, the cloud offloading decisions of all UEs
$\hat{\boldsymbol{\lambda}}_o$	$= (\hat{\lambda}_{o,1}, \hat{\lambda}_{o,2}, \ldots, \hat{\lambda}_{o,\lvert\mathcal{N}\rvert})^T$, the best cloud offloading decision set for all UEs
$\hat{K}$	all possible cloud offloading strategy sets of UEs

UE_n [25]. Moreover, each UE has a task to be completed. The task of UE_n is further defined as $a_n \triangleq (\delta_n, \omega_n, \zeta_n, d_n)$. For a_n, δ_n is the data size of a_n, which is measured by the number of bits. ω_n represents the number of CPU cycles needed to complete a_n. $\zeta_n \in \mathcal{I}$ is the location of UE_n when a_n is executed. d_n denotes the maximum service delay that UE_n can tolerate, i.e., the QoS requirement of the UE. ES_m is specified by $\tilde{f}_m$, $\tilde{r}_m$, and $\tilde{\zeta}_m$, where $\tilde{f}_m$ is the computing capability of the ES, which is quantified by the number of CPU cycles per second. $\tilde{r}_m$ is the data transmission rate of ES_m. $\tilde{\zeta}_m \in \mathcal{I}$ represents the location of ES_m. The computing capability of the cloud is denoted as $\hat{f}$, which is quantified by the number of CPU cycles per second. We use $\text{SN}_i \triangleq \{\text{UE}_n | \zeta_n = i\}$ and $\text{SM}_i \triangleq \{\text{ES}_m | \tilde{\zeta}_m = i\}$ to denote the set of UEs and the set of ESs in the ith service area, respectively. Moreover, we assume that ESs within the same service area are the same, i.e., $\tilde{f}_m = \tilde{f}_{m'}$ and $\tilde{r}_m = \tilde{r}_{m'}$ for all ES_m, $\text{ES}_{m'} \in \text{SM}_i$ [26]. However, ESs in different service areas are heterogeneous. Furthermore, we use $\lambda_n \in \{0,1\}$, $\tilde{\lambda}_{n,m} \in \{0,1\}$, and $\hat{\lambda}_n \in \{0,1\}$ to represent the offloading decision of a_n. If a_n is executed locally, $\lambda_n = 1$. Otherwise, $\lambda_n = 0$. If a_n is offloaded to ES_m, $\tilde{\lambda}_{n,m} = 1$. Otherwise, $\tilde{\lambda}_{n,m} = 0$. Similarly, $\hat{\lambda}_n = 1$ means that a_n is executed by the cloud. Otherwise, $\hat{\lambda}_n = 0$. In Hi-EECC, for $\text{UE}_n \in \text{SN}_i$, since a_n can be executed by one entity at a time, we have the following two constraints:

$$1 = \lambda_n + \sum_{m=1}^{|\mathcal{M}|} \tilde{\lambda}_{n,m} = \lambda_n + \sum_{\text{ES}_m \in \text{SM}_i} \tilde{\lambda}_{n,m}. \tag{1}$$

Accordingly, in Ho-EECC, we have

$$1 = \lambda_n + \sum_{m=1}^{|\mathcal{M}|} \tilde{\lambda}_{n,m} + \hat{\lambda}_n = \lambda_n + \sum_{\text{ES}_m \in \text{SM}_i} \tilde{\lambda}_{n,m} + \hat{\lambda}_n. \tag{2}$$

3.2 Computation Model

3.2.1 Local Computation Model

The computing capability of UE_n is quantified by the number of CPU cycles per second, i.e., f_n. Thus, the local computation delay of a_n is

$$t_n = \frac{\omega_n}{f_n}. \tag{3}$$

The energy consumption for UE_n executing a_n is calculated by the classic model used in [27], [28], i.e.,

$$e_n = \sigma_n t_n, \tag{4}$$

where σ_n can be obtained through the measurement approach [29], [30].

3.2.2 Edge Computation Model

The computing capability (i.e., computing resources) of ES_m is represented by $\tilde{f}_m$ and will be distributed proportionally to all UEs that request the ES. In this paper, the computing resource of ES_m allocated to UE_n is

$$f_{n,m} = \frac{\gamma_n}{\sum_{v=1}^{|\mathcal{N}|} \gamma_v \tilde{\lambda}_{v,m}} \tilde{f}_m = \frac{\gamma_n}{\sum_{\text{UE}_v \in \text{SN}_i} \gamma_v \tilde{\lambda}_{v,m}} \tilde{f}_m, \tag{5}$$

where $v \in [1, |\mathcal{N}|]$, $\sum_{\text{UE}_v \in \text{SN}_i} \gamma_v \tilde{\lambda}_{v,m}$ is the sum of resource weight parameters of all UEs that request ES_m and $\gamma_n / (\sum_{\text{UE}_v \in \text{SN}_i} \gamma_v \tilde{\lambda}_{v,m})$ is the resource proportion that UE_n can obtain from ES_m. According to the above equation, the computation delay of a_n executed by ES_m is formulated as

$$\tilde{t}_{n,m} = \frac{\omega_n}{f_{n,m}}. \tag{6}$$

UE_n focuses on minimizing its energy consumption and does not care about the cost of ESs. Therefore, the energy consumption of UE_n is zero when ES_m is executing a_n.

3.2.3 Cloud Computation Model

Compared with ESs, the cloud has sufficient resources to respond to the requests of UEs. Thus, we assume that the cloud server can handle an infinite number of tasks in parallel. The computation delay of a_n executed by the cloud can be formulated as

$$\hat{t}_n = \frac{\omega_n}{\hat{f}}. \tag{7}$$

Similarly, the energy consumption of UE_n is zero when the cloud is processing a_n.

3.3 Communication Model

3.3.1 Communication Model between UE_n and ES_m

Similar to the computing resource allocation policy, the communication resources (i.e., the data transmission rate) of ES_m are distributed proportionally to all UEs that request the ES. Thus, the data transmission rate of ES_m allocated to UE_n is

$$r_{n,m} = \frac{\gamma_n}{\sum_{v=1}^{|\mathcal{N}|} \gamma_v \tilde{\lambda}_{v,m}} \tilde{r}_m = \frac{\gamma_n}{\sum_{\text{UE}_v \in \text{SN}_i} \gamma_v \tilde{\lambda}_{v,m}} \tilde{r}_m. \tag{8}$$

Based on the above equation, the communication delay of a_n between UE_n and ES_m can be formulated as

$$\tilde{t}'_{n,m} = \frac{\delta_n}{r_{n,m}}. \tag{9}$$

The energy consumption of UE_n offloading a_n to ES_m is

$$\tilde{e}_{n,m} = \sigma'_n \tilde{t}'_{n,m}. \tag{10}$$

where σ'_n can be obtained by the long-term experience [31].

3.3.2 Communication Model between UE_n and the Cloud

The service mode of the cloud depends on the computing architecture type of EECC. Therefore, we formulate the communication models between a UE and the cloud in Hi-EECC and in Ho-EECC, respectively.

In Hi-EECC, UE_n cannot directly request the cloud. ES_m requested by UE_n decides whether to offload a_n to the cloud. In addition, a high-speed fiber communication link between ESs and the cloud is a necessary infrastructure in Hi-EECC. It ensures the flexibility and scalability of ESs, thereby providing UEs with high-quality services. Thus, we assume that the data transmission rate between ESs and the cloud is the same and is represented by $\hat{r}$. The communication delay for ES_m offloading a_n to the cloud can be formulated as

$$\hat{t}'_{n,hi} = \frac{\delta_n}{\hat{r}}. \tag{11}$$

Therefore, the transmission latency between UE_n and the cloud is $\tilde{t}'_{n,m} + \hat{t}'_{n,hi}$.

In Ho-EECC, UE_n can directly request the cloud. Moreover, the communication latency between UE_n and the cloud consists of two parts, i.e., the wireless communication delay and the wired communication delay [27]. We assume that the communication resources of the cloud are sufficient, i.e., the data transmission rate allocated to each UE is the same. Thus, the communication delay of a_n between UE_n and the cloud is

$$\hat{t}'_{n,ho} = \frac{\delta_n}{\hat{r}'_1} + \frac{\delta_n}{\hat{r}'_2}, \tag{12}$$

where $\hat{r}'_1$ and $\hat{r}'_2$ are the wireless data transmission rate and wired data transmission rate, respectively. Accordingly, the energy consumption of UE_n offloading a task to the cloud can be formulated as

$$\hat{e}_n = \sigma'_n \hat{t}'_{n,ho}. \tag{13}$$

3.4 Cost Model

Based on the previous definitions, for $\mathrm{UE}_n \in \mathrm{SN}_i$, the delay of a_n in Hi-EECC can be formulated as

$$\begin{aligned} T_{n,hi} = \lambda_n t_n + &\sum_{\mathrm{ES}_m \in \mathrm{SM}_i} \tilde{\lambda}_{n,m}(\tilde{t}_{n,m} + \tilde{t}'_{n,m}) \\ &+ \hat{\lambda}_n(\tilde{t}'_{n,m} + \hat{t}'_{n,hi} + \hat{t}_n). \end{aligned} \tag{14}$$

The energy consumption of UE_n in Hi-EECC is

$$E_{n,hi} = \lambda_n e_n + \sum_{\mathrm{ES}_m \in \mathrm{SM}_i} \tilde{\lambda}_{n,m}\tilde{e}_{n,m} + \hat{\lambda}_n \tilde{e}_{n,m}. \tag{15}$$

As shown in the above equation, the energy consumption for UE_n uploading tasks to the cloud is also $\tilde{e}_{n,m}$. The reason is that when ESs upload tasks to the cloud, the UE does not incur any energy consumption. The delay of a_n in Ho-EECC can be formulated as

$$\begin{aligned} T_{n,ho} = \lambda_n t_n + &\sum_{\mathrm{ES}_m \in \mathrm{SM}_i} \tilde{\lambda}_{n,m}(\tilde{t}_{n,m} + \tilde{t}'_{n,m}) \\ &+ \hat{\lambda}_n(\hat{t}_n + \hat{t}'_{n,ho}). \end{aligned} \tag{16}$$

The energy consumption of UE_n in Ho-EECC is

$$E_{n,ho} = \lambda_n e_n + \sum_{\mathrm{ES}_m \in \mathrm{SM}_i} \tilde{\lambda}_{n,m}\tilde{e}_{n,m} + \hat{\lambda}_n \hat{e}_n. \tag{17}$$

In this paper, the cost of UE_n is formulated as a weighted sum of the energy consumption of UE_n and the time consumption of a_n. Therefore, the cost of UE_n in Hi-EECC is

$$C_{n,hi} = \phi_n T_{n,hi} + (1-\phi_n)E_{n,hi}, \tag{18}$$

where $0 \leq \phi_n \leq 1$ is the weighted parameter of a_n's delay, which can represent the individual preference for energy consumption and time consumption. Similarly, the cost of UE_n in Ho-EECC is formulated as

$$C_{n,ho} = \phi_n T_{n,ho} + (1-\phi_n)E_{n,ho}. \tag{19}$$

4 A Potential Game Formulation

Due to the limited resources of ESs, there is a competitive relationship between UEs. All UEs have their own preferences and attempt to determine the most beneficial strategy for themselves. It is a considerable challenge to satisfy all UEs with a centralized method. Fortunately, game theory provides an efficient way to resolve the issue. Next, we construct a game for UEs and ESs, in which each UE selfishly minimizes its energy consumption and time consumption.

In EECC, there are $|\mathcal{N}|$ players (i.e., UEs) in a game and all UEs seek to minimize their cost. The nth player derives a strategy between itself and ESs, i.e., $\boldsymbol{\lambda_n} = (\lambda_n, \tilde{\lambda}_{n,1}, \ldots, \tilde{\lambda}_{n,|\mathcal{M}|}) \in K_n \subseteq \mathbb{R}^{|\mathcal{M}|+1}$, where K_n is all possible offloading strategies of UE_n. We use Λ to represent the offloading strategy set of all UEs, i.e., $\Lambda = (\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, \ldots, \boldsymbol{\lambda_{|\mathcal{N}|}})^T \in \mathcal{K} = K_1 \times K_2 \times \ldots \times K_{|\mathcal{N}|}$, where $\mathcal{K}$ is all possible offloading strategy sets of UEs. Λ_{-n} represents the offloading strategy set of $|\mathcal{N}|-1$ UEs except for UE_n, i.e., $\Lambda_{-n} = (\boldsymbol{\lambda_1}, \ldots, \boldsymbol{\lambda_{n-1}}, \boldsymbol{\lambda_{n+1}}, \ldots, \boldsymbol{\lambda_{|\mathcal{N}|}})^T$. Each UE has a payoff function $C(\boldsymbol{\lambda_n}, \Lambda_{-n}) \in \mathbb{R}$ in EECC game, where $C(\boldsymbol{\lambda_n}, \Lambda_{-n})$ represents the cost of UE_n adopting $\boldsymbol{\lambda_n}$ when Λ_{-n} is given. The game is called the EECC game.

Before offloading a task to ES_m, UE_n should ensure that its cost can be reduced. That is, UE_n should first assess the feasibility of $\mathrm{ES}_m \in \mathrm{SM}_i$. The feasibility of ES_m for UE_n can be assessed by the following theorem.

Theorem 1. *If $UE_n \in \mathrm{SN}_i$ offloads its task to $ES_m \in \mathrm{SM}_i$, that is, ES_m is an available ES for UE_n, then*

$$\sum_{v \neq n}^{|\mathcal{N}|} \gamma_v \tilde{\lambda}_{v,m} = \sum_{\mathrm{UE}_v \in \mathrm{SN}_i - \{\mathrm{UE}_n\}} \gamma_v \tilde{\lambda}_{v,m} \leq B_n, \tag{20}$$

where $B_n = (b_n - 1)\gamma_n$, and

$$b_n = \frac{\tilde{f}_m \tilde{r}_m \omega_n \big(\phi_n + (1-\phi_n)\sigma_n\big)}{f_n \phi_n(\omega_n \tilde{r}_m + \delta_n \tilde{f}_m) + (1-\phi_n)\delta_n f_n \tilde{f}_m \sigma'_n}. \tag{21}$$

Proof. If UE_n offloads its task to ES_m, then $\tilde{C}_{n,m} \leq C_{n,l}$, where

$$\tilde{C}_{n,m} = \phi_n(\tilde{t}_{n,m} + \tilde{t}'_{n,m}) + (1-\phi_n)\tilde{e}_{n,m}, \tag{22}$$

and

$$C_{n,l} = \phi_n t_n + (1-\phi_n)e_n. \tag{23}$$

Plugging Equations (3), (4), (5), (6), (9), and (10) into $\tilde{C}_{n,m} \leq C_{n,l}$, we have

$$\begin{aligned} &\phi_n\left(\frac{\omega_n}{f_{n,m}} + \frac{\delta_n}{r_{n,m}}\right) + (1-\phi_n)\sigma'_n \frac{\delta_n}{r_{n,m}} \\ &\leq \phi_n \frac{\omega_n}{f_n} + (1-\phi_n)\sigma_n \frac{\omega_n}{f_n}. \end{aligned} \tag{24}$$

That is,

$$\phi_n\left(\frac{\omega_n \sum_{v=1}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m}}{\gamma_n\tilde{f}_m}+\frac{\delta_n \sum_{v=1}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m}}{\gamma_n\tilde{r}_m}\right) +(1-\phi_n)\sigma'_n\frac{\delta_n \sum_{v=1}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m}}{\gamma_n\tilde{r}_m} \le \phi_n\frac{\omega_n}{f_n}+(1-\phi_n)\sigma_n\frac{\omega_n}{f_n}. \tag{25}$$

Rearranging the above inequality, we have

$$\frac{\sum_{v=1}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m}}{\gamma_n}\frac{\phi_n\omega_n\tilde{r}_m+\delta_n\phi_n\tilde{f}_m+(1-\phi_n)\delta_n\sigma'_n\tilde{f}_m}{\tilde{f}_m\tilde{r}_m} \le \frac{\phi_n\omega_n+(1-\phi_n)\omega_n\sigma_n}{f_n}, \tag{26}$$

i.e.,

$$\frac{\sum_{v=1}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m}}{\gamma_n}\le b_n, \tag{27}$$

where

$$b_n=\frac{\tilde{f}_m\tilde{r}_m\omega_n(\phi_n+(1-\phi_n)\sigma_n)}{f_n\phi_n(\omega_n\tilde{r}_m+\delta_n\tilde{f}_m)+(1-\phi_n)\delta_n f_n\tilde{f}_m\sigma'_n}.$$

Rearranging the above inequality, we can easily obtain

$$\sum_{v\neq n}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m}=\sum_{\mathrm{UE}_v\in \mathrm{SN}_i-\{\mathrm{UE}_n\}}\gamma_v\tilde{\lambda}_{v,m}\le(b_n-1)\gamma_n. \tag{28}$$

Thus, we have the theorem. □

Next, we provide the definition of the potential game [32]. Then, we introduce a potential function to transform the EECC game into a potential game [33].

Definition 1. *A game is called a potential game, if there is a potential function $\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n})$ that satisfies*

$$C(\boldsymbol{\lambda_n},\Lambda_{-n}) < C(\boldsymbol{\lambda'_n},\Lambda_{-n}) \Leftrightarrow \Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n}) < \Phi_{\Lambda_{-n}}(\boldsymbol{\lambda'_n}),$$

for $UE_n\in\mathcal{N}$, $\boldsymbol{\lambda_n}\in K_n$, $\Lambda\in\mathcal{K}$, and $\Lambda_{-n}\in\Pi_{v\neq n}K_v$. $\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n})$ is a potential function of $\boldsymbol{\lambda_n}$ when Λ_{-n} is given.

As shown in Equations (5), (6), (8), (9), (10), and (14)-(19), when the offloading strategies of $|\mathcal{N}|-1$ UEs except for UE_n are given, the cost of UE_n depends on the ES it chooses and its γ_n. Since we assume that ESs within the same service area are the same, if $\tilde{\lambda}_{n,m}=1$, the number and types of requests responded by ES_m (i.e., $\sum_{n=1}^{|\mathcal{N}|}\gamma_n\tilde{\lambda}_{n,m}$) determine the cost of the UE. Based on Theorem 1, we construct a potential function for the EECC game in the following theorem.

Theorem 2. *If all ESs in the ith service area are the same, i.e., $\tilde{f}_m=\tilde{f}_{m'}$ and $\tilde{r}_m=\tilde{r}_{m'}$ for all ES_m, $\mathrm{ES}_{m'}\in \mathrm{SM}_i$, then the EECC game is a potential game with the following potential function:*

$$\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n})=\frac{1}{2}\sum_{n=1}^{|\mathcal{N}|}\sum_{v\neq n}^{|\mathcal{N}|}\sum_{m=1}^{|\mathcal{M}|}\gamma_n\gamma_v\tilde{\lambda}_{n,m}\tilde{\lambda}_{v,m} +\sum_{n=1}^{|\mathcal{N}|}\gamma_n B_n\lambda_n. \tag{29}$$

Proof. According to the definition of a potential game, we should prove that the potential function increases or decreases with an increase or decrease in $C(\boldsymbol{\lambda_n},\Lambda_{-n})$. To demonstrate the above property of $\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n})$, we consider the following three cases. Let $\boldsymbol{\lambda_n}=(\lambda_n,\tilde{\lambda}_{n,1},\ldots,\tilde{\lambda}_{n,|\mathcal{M}|})$ and $\boldsymbol{\lambda'_n}=(\lambda'_n,\tilde{\lambda}'_{n,1},\ldots,\tilde{\lambda}'_{n,|\mathcal{M}|})$ be two offloading strategies of UE_n, where $\boldsymbol{\lambda_n}\neq\boldsymbol{\lambda'_n}$.

Case 1: Suppose that $\tilde{\lambda}_{n,m}=1$, $\tilde{\lambda}'_{n,m'}=1$, and $\tilde{C}_{n,m'}<\tilde{C}_{n,m}$, where $m\neq m'$. We know that the UEs that initiate requests to the same ES can affect each other. Moreover, since adjusting the offloading strategy among ESs does not affect other UEs that perform their tasks locally, we have $\sum_{n=1}^{|\mathcal{N}|}\gamma_n B_n\lambda_n=\sum_{n=1}^{|\mathcal{N}|}\gamma_n B_n\lambda'_n$. Based on Equation (29), since $\tilde{\lambda}_{n,m}=1$, $\tilde{\lambda}'_{n,m'}=1$, and $\lambda_n+\sum_{m=1}^{|\mathcal{M}|}\tilde{\lambda}_{n,m}=1$, we have

$$\begin{aligned}
&\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n})-\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda'_n})\\
&=\frac{1}{2}\gamma_n\tilde{\lambda}_{n,m}\sum_{v\neq n}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m}+\frac{1}{2}\gamma_n\tilde{\lambda}_{n,m}\sum_{v\neq n}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m}\\
&\quad+\frac{1}{2}\sum_{v'\neq n}^{|\mathcal{N}|}\gamma_{v'}\tilde{\lambda}_{v',m}\sum_{v\neq n,v\neq v'}^{|\mathcal{N}|}\sum_{m=1}^{|\mathcal{M}|}\gamma_v\tilde{\lambda}_{v,m}\\
&\quad-\frac{1}{2}\sum_{v'\neq n}^{|\mathcal{N}|}\gamma_{v'}\tilde{\lambda}_{v',m'}\sum_{v\neq n,v\neq v'}^{|\mathcal{N}|}\sum_{m'=1}^{|\mathcal{M}|}\gamma_v\tilde{\lambda}_{v,m'}\\
&\quad-\frac{1}{2}\gamma_n\tilde{\lambda}'_{n,m'}\sum_{v\neq n}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m'}-\frac{1}{2}\gamma_n\tilde{\lambda}'_{n,m'}\sum_{v\neq n}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m'}\\
&=\gamma_n\sum_{v\neq n}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m}-\gamma_n\sum_{v\neq n}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m'}.
\end{aligned} \tag{30}$$

If $\tilde{C}_{n,m'}<\tilde{C}_{n,m}$, based on Equation (22), we have

$$\phi_n(\tilde{t}_{n,m'}+t'_{n,m'})+(1-\phi_n)e_{n,m'} < \phi_n(\tilde{t}_{n,m}+\tilde{t}'_{n,m})+(1-\phi_n)\tilde{e}_{n,m}. \tag{31}$$

Plugging Equations (5), (6), and (10) into the above inequality, we have

$$\phi_n\left(\frac{\omega_n}{f_{n,m'}}+\frac{\delta_n}{r_{n,m'}}\right)+(1-\phi_n)\sigma'_n\frac{\delta_n}{r_{n,m'}} < \phi_n\left(\frac{\omega_n}{f_{n,m}}+\frac{\delta_n}{r_{n,m}}\right)+(1-\phi_n)\sigma'_n\frac{\delta_n}{r_{n,m}}. \tag{32}$$

Rearranging the above inequality, we have

$$\frac{\sum_{v=1}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m'}}{\gamma_n}\frac{\phi_n\omega_n\tilde{r}_{m'}+\delta_n\phi_n\tilde{f}_{m'}+(1-\phi_n)\delta_n\sigma'_n\tilde{f}_{m'}}{\tilde{f}_{m'}\tilde{r}_{m'}} < \frac{\sum_{v=1}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m}}{\gamma_n}\frac{\phi_n\omega_n\tilde{r}_m+\delta_n\phi_n\tilde{f}_m+(1-\phi_n)\delta_n\sigma'_n\tilde{f}_m}{\tilde{f}_m\tilde{r}_m}. \tag{33}$$

If all ESs in the ith service area are the same, we have $\tilde{f}_m=\tilde{f}_{m'}$ and $\tilde{r}_m=\tilde{r}_{m'}$, i.e.,

$$\sum_{v=1}^{|\mathcal{N}|} \gamma_v \tilde{\lambda}_{v,m'} < \sum_{v=1}^{|\mathcal{N}|} \gamma_v \tilde{\lambda}_{v,m}. \tag{34}$$

Since $\gamma_n > 0$, it can be easily found that

$$\gamma_n \sum_{v \neq n}^{|\mathcal{N}|} \gamma_v \tilde{\lambda}_{v,m'} < \gamma_n \sum_{v \neq n}^{|\mathcal{N}|} \gamma_v \tilde{\lambda}_{v,m}. \tag{35}$$

Therefore, $\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n}) - \Phi_{\Lambda_{-n}}(\boldsymbol{\lambda'_n}) > 0$.

Case 2: Suppose that $\lambda_n = 1$, $\tilde{\lambda}'_{n,m'} = 1$, and $\tilde{C}_{n,m} < C_{n,l}$. Based on Equation (29), we have

$$\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n}) - \Phi_{\Lambda_{-n}}(\boldsymbol{\lambda'_n}) = \gamma_n \sum_{v \neq n}^{|\mathcal{N}|} \gamma_v \tilde{\lambda}'_{v,m} - \gamma_n B_n \lambda_n. \tag{36}$$

Based on Theorem 1, we obtain $\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n}) - \Phi_{\Lambda_{-n}}(\boldsymbol{\lambda'_n}) > 0$.

Case 3: Suppose that $\tilde{\lambda}_{n,m} = 1$, $\lambda'_n = 1$, and $C_{n,l} < \tilde{C}_{n,m'}$. Similar to case 2, we can also easily obtain that $\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n})$ increases or decreases with the increase or decrease in $C(\boldsymbol{\lambda_n}, \Lambda_{-n})$. □

Remark 1. In this paper, we construct a potential function for the EECC game, but do not formulate specific potential functions for either Hi-EECC or Ho-EECC. On the one hand, the design is restricted by the potential game theory. Potential game theory requires that all servers be homogeneous [32]. However, although we assume that ESs within the same service area are the same, ESs in different service areas are heterogeneous. In addition, there are huge differences between the cloud and ESs.

On the other hand, the design takes into account the characteristics of Hi-EECC and Ho-EECC. As mentioned above, the way that the cloud responds to UEs is determined by the computing architectures. For $\text{UE}_n \in \text{SN}_i$, UE_n can temporarily ignore the existence of the cloud and determine a preliminary strategy between itself and ESs (i.e., all $\text{ES}_m \in \text{SM}_i$). In Hi-EECC, the deadline unsatisfied task of UE_n is further uploaded to the cloud by the ESs, the final best strategy that satisfy its QoS demand can be obtained. In Ho-EECC, by comparing the preliminary strategy with the strategy of cloud processing, the final best strategy with less cost can be derived.

In the EECC game, the nth player derives a strategy between itself and ESs, i.e., $\boldsymbol{\lambda_n} = (\lambda_n, \tilde{\lambda}_{n,1}, \ldots, \tilde{\lambda}_{n,|\mathcal{M}|})$. Since UEs can initiate the requests only to the ESs that in the same service area as the UEs, and the ESs within the same service area are the same, we can construct a potential function, i.e., Equation (29), to transform the EECC game into a potential game. Thus, regardless of the specific computing architectures, according to Theorem 2, we can develop Algorithm 1 to determine the preliminary strategies of all UEs. Then, based on the characteristics of Hi-EECC and Ho-EECC, we can further develop different algorithms (i.e., COAHi and COAHo) to readjust the offloading strategies obtained from Algorithm 1 to obtain the final best strategies for all UEs, so that the potential game theory can solve the strategy optimization problem in the heterogeneous scenario.

It should also be noted that a potential game may have many potential functions. However, for a potential game, different potential functions do not affect the quality of the strategy [32], [33]. Therefore, we do not formulate other potential functions or explore the impact of these functions on the performance of the proposed algorithms and two computing architectures.

Since there are competitive relationships between UEs, the strategy of a UE affects the cost of other UEs. Thus, we must determine a best strategy set that can be accepted by all UEs, i.e., Nash equilibrium. We now present the definition of Nash equilibrium.

Definition 2. *A strategy set $\Lambda^* = (\boldsymbol{\lambda}_1^*, \boldsymbol{\lambda}_2^*, \ldots, \boldsymbol{\lambda}_{|\mathcal{N}|}^*)^T$ is a Nash equilibrium of the EECC game, i.e., no UE can unilaterally change its strategy to further reduce its cost, if*

$$C(\boldsymbol{\lambda_n^*}, \Lambda^*_{-n}) \leq C(\boldsymbol{\lambda_n}, \Lambda^*_{-n}), \text{ for all } \boldsymbol{\lambda_n} \in K_n, \tag{37}$$

holds for all $\text{UE}_n \in \mathcal{N}$.

As shown in Definition 2, Nash equilibrium is the state in which all UEs find the best offloading strategies toward each other. It should be noted that not every game has a Nash equilibrium. Fortunately, if a game can be formulated as a potential game, there is at least one Nash equilibrium of the game [33]. Moreover, according to the finite improvement property, the Nash equilibrium of the game can be obtained after a finite number of iterations [32]. This motivates us to develop an iteration algorithm to find the Nash equilibrium of the EECC game. We present the algorithm in Section 5.1 and analyze the finite improvement property in Section 6.1.

5 Potential Game-Based Algorithms in Hi-EECC and Ho-EECC

5.1 Algorithms in Hi-EECC

According to Theorem 2, we develop an iteration offloading algorithm, i.e., Algorithm 1, for optimizing the offloading decisions of UEs between itself and ESs. We first initialize the strategies of UEs, i.e., $\boldsymbol{\lambda_n} = (1, 0, \ldots, 0)_{|\mathcal{M}|+1}$, for all $\text{UE}_n \in \mathcal{N}$ (Line 1). Then, we can calculate the initial potential function of each UE (Line 3). Next, we iterate every UE and make a new offloading decision with less cost (Lines 4-19). If no UE can unilaterally change its strategy to further reduce its cost, the game is over, i.e., the final strategy set Λ^* is regarded as a Nash equilibrium (Lines 21-25). Moreover, to control the time complexity of the algorithm, we can define the maximum iteration number Π to limit the number of iterations, thus obtaining an acceptable strategy set of UEs. In Section 6.1, Theorem 4 analyzes the convergence of the algorithm in detail.

In Hi-EECC, UE_n makes offloading decisions depending on whether its cost can be reduced. However, if $\tilde{\lambda}_{n,m} = 1$ and the delay served by ES_m exceeds d_n, the ES will offload a_n to the cloud for executing. It incurs the cost of ES_m for the ES uploading tasks to the cloud. In this paper, we define the cost of ES_m as the price paid for the time required by the cloud to complete the tasks. The cost of ES_m is

$$C_{m,c}(\Lambda) = \sum_{n=1}^{|\mathcal{N}|} \frac{\omega_n}{\hat{f}} \left(1 - \sum_{m=1}^{|\mathcal{N}|} \tilde{\lambda}_{n,m} - \lambda_n\right) \kappa_m, \tag{38}$$

where κ_m is the correlation parameter between the computation delay for the cloud completing a_n and ES$_m$'s cost.

Algorithm 1. Nash Equilibrium Calculating Algorithm

Input: $\gamma_n, \omega_n, \zeta_n, d_n, \delta_n, \sigma_n, \sigma'_n$, and f_n, for all UE$_n \in \mathcal{N}$. $\tilde{f}_m$, $\tilde{r}_m$, and κ_m for all ES$_m \in \mathcal{M}$. $\hat{f}$, $\hat{r}$, and Π.
Output: Λ^*.

```
Initialize Λ ← ((1,0,...,0),...,(1,0,...,0))^T_{|N|};
while π < Π do
    Calculate Φ_{Λ_{-n}}(λ_n) for all UE_n ∈ N based on Equation (29);
    for UE_n ∈ N do
        λ_n ← (1,0,...,0)_{|M|+1};
        i ← ζ_n;
        for ES_m ∈ SM_i do
            if ES_m is an available ES for UE_n then
                λ'_n ← (0,...,0)_{|M|+1};
                λ'_{n,m+1} ← 1;
                Calculate Φ_{Λ_{-n}}(λ'_n) based on Equation (29);
                if Φ'_Λ(n) < Φ_Λ(n) then
                    λ_n ← λ'_n;
                    Φ_{Λ_{-n}}(λ_n) ← Φ_{Λ_{-n}}(λ'_n);
                end if
            end if
        end for
        Update the offloading strategy of UE_n between itself and ESs, i.e., λ*_n ← λ_n;
    end for
    π ← π + 1
    if no UE can unilaterally change its strategy to further reduce its cost then
        break;
    end if
end while
return Λ* = (λ*_1, λ*_2, ..., λ*_{|N|})^T.
```

Let us suppose that UE$_n$ and UE$_v$ request ES$_m$ for processing their tasks. Furthermore, the deadlines of a_n and a_v cannot be satisfied by ES$_m$. Thus, the two tasks could be uploaded to the cloud. However, while the ES uploads a_n to the cloud, the resources originally allocated to UE$_n$ will be released. The released resources can provide service to UE$_v$ and thus may satisfy the QoS demand of UE$_v$. Therefore, the ES should also optimize the tasks that are uploaded to the cloud to reduce its cost. The optimization objective of ES$_m$ can be formulated as the following problem:

$$\begin{aligned} &\mathrm{P1}: \min_{\Lambda} C_{m,c}(\Lambda) \\ &s.t.\ C1: T_{n,hi} \le d_n,\ \text{for all UE}_n \in \mathcal{N}, \\ &C2: \sum_{m=1}^{|\mathcal{M}|} \tilde{\lambda}_{n,m} + \hat{\lambda}_n = 1,\ \text{where } \tilde{\lambda}_{n,m}, \hat{\lambda}_n \in \{0,1\}, \end{aligned} \tag{39}$$

where $C1$ ensures that QoS demands of all UEs should be satisfied. $C2$ means that a task can be executed by only one entity. It is clear that P1 is an NP-hard problem [34].

Remark 2. In Hi-EECC, when a_n is executed by the cloud, the cost of UE$_n$ is

$$\hat{C}_n = \phi_n(\hat{t}_n + \hat{t}'_{n,hi} + \tilde{t}'_{n,m}) + (1 - \phi_n)\tilde{e}_{n,m}. \tag{40}$$

Based on Equations (22) and (40), if offloading a_n to the cloud, the following inequality should be true:

$$\tilde{C}_{n,m} - \hat{C}_n = \phi_n(\tilde{t}_{n,m} - \hat{t}_n - \hat{t}'_{n,hi}) \ge 0. \tag{41}$$

Otherwise, the QoS demand of UE$_n$ cannot be met. Therefore, this paper has an implicit requirement that the cloud has sufficient resources to meet the demands of UEs, i.e., $\hat{f} \gg \tilde{f}_m$. Hence, for a_n, if $\tilde{\lambda}_{n,m} = 1$ and $T_{n,hi} > d_n$, offloading the task to the cloud by ES$_m$ should not increase the cost of UE$_n$.

Moreover, if a_n is offloaded to the cloud, we have

$$\sum_{n=1}^{|\mathcal{N}|} \gamma_n \tilde{\lambda}_{n,m} > \sum_{v \ne n}^{|\mathcal{N}|} \gamma_v \tilde{\lambda}_{v,m}. \tag{42}$$

According to Equations (6), (9), (10), and (18), we can easily determine that if a_n is offloaded to the cloud, the cost of other UEs that request ES$_m$ decreases.

Algorithm 2. Deadline Guaranteeing Algorithm

Input: The offloading strategy set Λ obtained from Algorithm 1. $\hat{\boldsymbol{\lambda}} = (0,\ldots,0)^T_{|\mathcal{N}|}$. $\gamma_n, \omega_n, \zeta_n, d_n, \delta_n, \sigma_n, \sigma'_n$, and f_n, for all UE$_n \in \mathcal{N}$. $\tilde{f}_m$, $\tilde{r}_m$, and κ_m for all ES$_m \in \mathcal{M}$. $\hat{f}$, $\hat{r}$.
Output: New offloading strategy set Λ, $\hat{\boldsymbol{\lambda}}$, and the cost of ES$_m$ $C_{m,c}(\Lambda)$.

```
R ← {UE_n | for all UE_n ∈ N, where λ̃_{n,m} = 1 and T_{n,hi} > d_n};
while R ≠ ∅ do
    UE_n ← arg max_{UE_n ∈ R}{γ_n} or arg min_{UE_n ∈ R}{ω_n};
    Update the cloud offloading decision, i.e., λ̂_n ← 1;
    Update the offloading decision of UE_n between itself and ESs, i.e., λ_n;
    R ← R − {UE_n};
    Calculate T_{n,hi} for all UE_n ∈ R;
    R' ← {UE_n | λ̃_{n,m} = 1 and T_{n,hi} ≤ d_n};
    R ← R − R';
end while
Calculate C_{m,c}(Λ) based on Equation (38);
return Λ, λ̂, C_{m,c}(Λ).
```

According to Remark 2, ESs can safely upload tasks to the cloud. To solve P1, we propose Algorithm 2 based on the greedy policy. Specifically, the algorithm reschedules the tasks based on ω_n or γ_n. The reason is that uploading tasks with larger γ_n will increase the released resources such that more UEs' demands can be satisfied. In addition, uploading tasks with smaller ω_n will directly help reduce the cost of ES$_m$. We can obtain two task offloading rescheduling schemes by using the greedy policy. The scheme with the lowest cost is regarded as the final strategy. For simplicity, we use $\hat{\lambda}$ to represent the cloud offloading decisions of all UEs, i.e., $\hat{\lambda} = (\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_{|\mathcal{N}|})^T \in \hat{K} \subseteq \mathbb{R}^{|\mathcal{N}|}$, where $\hat{K}$ is all possible cloud offloading strategy sets of UEs.

Algorithm 2 shows the two rescheduling schemes based on different greedy policies. We first iterate every ES$_m \in \mathcal{M}$

and identify the deadline unsatisfied tasks (Line 1), i.e., $\mathcal{R}$. The task with the maximum γ_n among all $\mathrm{UE}_v \in \mathcal{R}$ is offloaded to the cloud (Lines 2-6). However, after a task is offloaded to the cloud, it is necessary to check whether the current resources can meet the QoS demands of other original deadline unsatisfied tasks. Thus, we update the computation and communication delay of the remaining tasks in $\mathcal{R}$ (Line 7), and remove the UEs whose demands can be satisfied from $\mathcal{R}$ (Lines 8-9). The above process is iteratively operated, until $\mathcal{R} = \emptyset$. $C_{m,c}(\Lambda)$ can be obtained after the rescheduling processing (Line 12). Similarly, we can obtain a rescheduling scheme based on $\mathrm{UE}_n \leftarrow \arg\min_{\mathrm{UE}_n \in \mathcal{R}}\{\omega_n\}$. It is easy to know that the time complexity of the algorithm is $O(|\mathcal{N}|)$.

Algorithm 3. Computation Offloading Algorithm in Hi-EECC (COAHi)

Input: $\gamma_n, \omega_n, \zeta_n, d_n, \delta_n, \sigma_n, \sigma'_n$, and f_n, for all $\mathrm{UE}_n \in \mathcal{N}$. $\tilde{f}_m$, $\tilde{r}_m$, and κ_m for all $\mathrm{ES}_m \in \mathcal{M}$. $\hat{f}$, $\hat{r}$.
Output: Λ_o, $\hat{\boldsymbol{\lambda}}_o$.

```
1: Obtain Λ through Algorithm 1;
2: Initialize the cloud strategies of UEs λ̂ = (0, ..., 0)^T_{|N|};
3: for ES_m ∈ M do
4:    Obtain Λ_1, λ̂_1, and C_1 through Algorithm 2 based on UE_n
      ← arg max_{UE_n ∈ R}{γ_n};
5:    Obtain Λ_2, λ̂_2, and C_2 through Algorithm 2 based on UE_n
      ← arg min_{UE_n ∈ R}{ω_n};
6:    if C_1 < C_2 then
7:       Update Λ and λ̂ according to Λ_1 and λ̂_1, respectively;
8:    else if C_1 ≥ C_2 then
9:       Update Λ and λ̂ according to Λ_2 and λ̂_2, respectively;
10:   end if
11: end for
12: Obtain the final strategies of UEs between itself and ESs,
    i.e., Λ_o ← Λ;
13: Obtain the final cloud offloading strategies of UEs, i.e.,
    λ̂_o ← λ̂;
14: return Λ_o, λ̂_o.
```

We first develop Algorithm 1 to determine a preliminary decision set between UEs and ESs. However, Algorithm 1 aims only at minimizing the cost of UEs, and does not consider QoS requirements of the UEs. The decisions obtained by Algorithm 1 may cause QoS requirements of some tasks to not be met. Thus, we then develop Algorithm 2 to determine whether to continue uploading these unsatisfied tasks to the cloud to meet their demands, that is, to solve P1. Algorithm 3 is developed based on Algorithms 1 and 2, and is the algorithm for making offloading strategies in Hi-EECC, which is named COAHi. The initial decision set between local processing and offloading to ESs is obtained from Algorithm 1 (Line 1). Then, we reschedule the tasks between ESs and the cloud by using Algorithm 2 (Lines 3-5). Finally, we update Λ and $\hat{\lambda}$ according to the rescheduling decision set with less cost and obtain the final offloading strategies of UEs (Lines 6-14). The time complexity of Algorithm 3 is derived in Corollary 1. It should be noted that since we readjust the decisions obtained by Algorithm 1, the original Nash equilibrium of UEs is broken. Based on Remarks 1 and 2, although the final offloading strategy set obtained by Algorithm 3 is not a Nash equilibrium of the EECC game, the set consists of the best strategies of all UEs.

5.2 Algorithms in Ho-EECC

In Ho-EECC, UEs can directly offload their tasks to the cloud. Thus, UE_n can first make an offloading decision by using Algorithm 1. Then, the cost of the decision is compared with the cost of cloud processing. Therefore, the final offloading strategy can be obtained. As mentioned above, in contrast to the decision-making method in Hi-EECC, UE_n directly determines the offloading strategy. Hence, before making a decision, the UE should not only ensure that its cost can be reduced, but also ensure that its QoS demand can be guaranteed. In addition to Theorem 1, the feasibility of ES_m for UE_n should be checked using the following theorem.

Algorithm 4. Computation Offloading Algorithm in Ho-EECC (COAHo)

Input: $\gamma_n, \omega_n, \zeta_n, d_n, \delta_n, \sigma_n, \sigma'_n$, and f_n, for all $\mathrm{UE}_n \in \mathcal{N}$. $\tilde{f}_m$, $\tilde{r}_m$, and κ_m for all $\mathrm{ES}_m \in \mathcal{M}$. $\hat{f}$, $\hat{r}_1$, $\hat{r}_2$, and Π.
Output: Λ_o, $\hat{\boldsymbol{\lambda}}_o$.

```
N' ← N;
Obtain Λ through Algorithm 1;
Initialize the cloud strategies of UEs λ̂ = (0, ..., 0)^T_{|N|};
for UE_n ∈ N' do
   Calculate C_{n,ho} based on Λ, λ̂, and Equation (19);
   λ̂_n ← 1;
   λ'_n ← (0, ..., 0)_{|M|+1};
   Calculate C'_{n,ho} based on Λ, λ̂, and Equation (19);
   if C'_{n,ho} < C_{n,ho} then
      λ̂_n ← 1;
      λ_n ← λ'_n;
      N' ← N' − {UE_n};
   end if
      Update λ_n for all UE_n ∈ N' through Algorithm 1, and
      obtain new offloading strategy set Λ;
end for
Obtain the final strategies of UEs between itself and ESs,
i.e., Λ_o ← Λ;
Obtain the final cloud offloading strategies of UEs, i.e.,
λ̂_o ← λ̂;
return Λ_o, λ̂_o.
```

Theorem 3. *$\mathrm{ES}_m \in \mathrm{SM}_i$ is an available ES for $\mathrm{UE}_n \in \mathrm{SN}_i$ when the ES satisfies Theorem 1 and the following inequality:*

$$\sum_{v \neq n}^{|\mathcal{N}|} \gamma_v \tilde{\lambda}_{v,m} \leq B'_n, \tag{43}$$

where

$$B'_n = \frac{d_n \gamma_n \tilde{f}_m \tilde{r}_m}{\omega_n \tilde{r}_m + \delta_n \tilde{f}_m} - \gamma_n. \tag{44}$$

Proof. If a_n is offloaded to ES_m for executing, the computation and communication delay of a_n should satisfy the following inequality:

$$\tilde{t}_{n,m} + \tilde{t}'_{n,m} \leq d_n. \tag{45}$$

Plugging Equations (6) and (9) into the above inequality, we have

$$\frac{\sum_{n=1}^{|\mathcal{N}|}\gamma_n\tilde{\lambda}_{n,m}}{\gamma_n}\left(\frac{\omega_n}{\tilde{f}_m}+\frac{\delta_n}{\tilde{r}_m}\right)\le d_n, \tag{46}$$

i.e.,

$$\sum_{v\ne n}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m}\le\frac{d_n\gamma_n\tilde{f}_m\tilde{r}_m}{\omega_n\tilde{r}_m+\delta_n\tilde{f}_m}-\gamma_n.$$

Thus, we reach the conclusion. □

Algorithm 4 is developed based on Algorithm 1, and describes the processing for making task offloading strategies for UEs in Ho-EECC, which is named COAHo. It should be noted that although Algorithms 3 and 4 both call Algorithm 1, the criteria for checking the availability of ESs is different. In Algorithm 3, the availability of ESs is checked by Theorem 1. In Algorithm 4, the availability of ESs is checked by Theorem 3. The initial strategy set Λ is obtained through Algorithm 1 (Line 2). Then, the current cost of UE$_n$ is compared with the cloud execution cost of the UE. If the cloud execution cost is less than the current decision cost, UE$_n$ will reschedule its task to the cloud (Lines 4-13). As some UEs are uploaded to the cloud, the resources originally occupied by these UEs are provided to other UEs. Therefore, the strategy should be updated again through Algorithm 1 (Line 14). If none of the UEs can benefit from the update process, the algorithm ends (Lines 16-18). Similar to COAHi, since we readjust the decisions obtained by Algorithm 1, the original Nash equilibrium of UEs is broken. Moreover, although the final offloading strategy set obtained by Algorithm 4 is not a Nash equilibrium of the EECC game, the set consists of the best strategies of all UEs.

6 Performance Analysis

6.1 Convergence of Algorithms

The finite improvement property of the potential game ensures that the game approaches a Nash equilibrium after the finite iteration [32]. Next, we analyze the convergence of the proposed algorithms. Let $\Gamma_{max}\triangleq\max\{\gamma_n|\text{ for all UE}_n\in\mathcal{N}\}$, $\Gamma_{min}\triangleq\min\{\gamma_n|\text{ for all UE}_n\in\mathcal{N}\}$, and $B_{max}=\max\{B_n|\text{ for all UE}_n\in\mathcal{N}\}$. Furthermore, γ_n and B_n are assumed to be non-negative integers.

Theorem 4. *For Algorithm 1, the maximum number of iterations for UE$_n$ determining an offloading strategy is*

$$\Pi_{max}\le\frac{|\mathcal{N}|^2\Gamma_{max}^2}{2\Gamma_{min}}+\frac{|\mathcal{N}|\Gamma_{max}B_{max}}{\Gamma_{min}}. \tag{47}$$

Proof. Based on Equation (29), we have

$$\begin{aligned}\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n})&\le\frac{1}{2}\sum_{n=1}^{|\mathcal{N}|}\sum_{n=1}^{|\mathcal{N}|}\Gamma_{max}^2+\sum_{n=1}^{|\mathcal{N}|}\Gamma_{max}B_{max}\\&\le\frac{1}{2}|\mathcal{N}|^2\Gamma_{max}^2+|\mathcal{N}|\Gamma_{max}B_{max}.\end{aligned} \tag{48}$$

Let $\boldsymbol{\lambda_n}=(\lambda_n,\tilde{\lambda}_{n,1},\ldots,\tilde{\lambda}_{n,|\mathcal{M}|})$ and $\boldsymbol{\lambda'_n}=(\lambda'_n,\tilde{\lambda}'_{n,1},\ldots,\tilde{\lambda}'_{n,|\mathcal{M}|})$ be two offloading strategies of UE$_n$, where $\boldsymbol{\lambda_n}\ne\boldsymbol{\lambda'_n}$. Then we prove that if UE$_n$ updates its strategy from $\boldsymbol{\lambda_n}$ to $\boldsymbol{\lambda'_n}$, we have

$$\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n})-\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda'_n})\ge\Gamma_{min}. \tag{49}$$

Case 1: we suppose that $\tilde{\lambda}_{n,m}=1$ and $\tilde{\lambda}'_{n,m'}=1$, where $m\ne m'$. According to Equations (30) and (35), we have

$$\begin{aligned}&\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n})-\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda'_n})\\&=\gamma_n\left(\sum_{v\ne n}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m}-\sum_{v\ne n}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m'}\right)>0.\end{aligned} \tag{50}$$

Since γ_n is assumed to be an integer, we have

$$\sum_{v\ne n}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m}-\sum_{v\ne n}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m'}\ge1. \tag{51}$$

It can be easily obtained that

$$\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n})-\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda'_n})>\Gamma_{min}. \tag{52}$$

Case 2: we suppose that $\lambda_n=1$ and $\tilde{\lambda}'_{n,m}=1$. According to Equations (36), we have

$$\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n})-\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda'_n})=\gamma_n\sum_{v\ne n}^{|\mathcal{N}|}\gamma_v\tilde{\lambda}_{v,m}-\gamma_nB_n>0. \tag{53}$$

Similarly, since γ_n is assumed to be an integer, we obtain

$$\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n})-\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda'_n})\ge1. \tag{54}$$

Accordingly, we also obtain $\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda_n})-\Phi_{\Lambda_{-n}}(\boldsymbol{\lambda'_n})\ge\Gamma_{min}$. Based on Equations (48) and (49), we know that the maximum number of iterations for a UE making an offloading strategy by using Algorithm 1 is

$$\Pi_{max}\le\frac{|\mathcal{N}|^2\Gamma_{max}^2}{2\Gamma_{min}}+\frac{|\mathcal{N}|\Gamma_{max}B_{max}}{\Gamma_{min}}.$$

Based on the above analysis, we have the theorem. □

According to Theorem 4, we can derive the time complexity of Algorithm 3, and have the following corollary.

Corollary 1. *Since Algorithm 3 calls Algorithm 1 once and Algorithm 2 twice, it can be easily derived that the time complexity of Algorithm 3 is $O(|\mathcal{N}|^2)$.*

Theorem 5. *For Algorithm 4, the maximum number of iterations for UE$_n$ making an offloading strategy is*

$$\Pi_{max}\le\frac{|\mathcal{N}|^3\Gamma_{max}^2}{2\Gamma_{min}}+\frac{|\mathcal{N}|^2\Gamma_{max}B_{max}}{\Gamma_{min}}. \tag{55}$$

Proof. Compared with Algorithm 1, after obtaining the offloading strategies between UEs and ESs, Algorithm 4 then reschedules the tasks between the ESs and the cloud to obtain the offloading strategies with less cost. Thus, in Algorithm 4, UE$_n$ performs Algorithm 1 no more than $|\mathcal{N}|$ times to make all offloading strategies of UEs. Based on Theorem 4, we reach the conclusion. □

6.2 Performance of Algorithms

Although performance evaluation is not the focus of game theory, it is interesting to investigate the performance of potential game-based algorithms. To analyze the performance of the algorithms proposed in this paper, we investigate the price of anarchy (PoA) in system-wide cost, which quantifies the efficiency ratio of the worst-case Nash equilibrium strategy over the optimal strategy obtained through the centralized methods [35]. In this paper, the system-wide cost of UEs is the total cost of all $\mathrm{UE}_n \in \mathcal{N}$, i.e., $\sum_{n=1}^{|\mathcal{N}|} C_n$ in Hi-EECC and $\sum_{n=1}^{|\mathcal{N}|} C'_n$ in Ho-EECC. In Hi-EECC, PoA is defined as

$$\mathrm{PoA} = \frac{\sum_{n=1}^{|\mathcal{N}|} C_n(\overline{\boldsymbol{\lambda}}_{\boldsymbol{n}})}{\max_{\boldsymbol{\lambda}_{\boldsymbol{n}}} \sum_{n=1}^{|\mathcal{N}|} C_n(\boldsymbol{\lambda}^*_{\boldsymbol{n}})}, \tag{56}$$

where $\overline{\boldsymbol{\lambda}}_{\boldsymbol{n}}$ is an optimal offloading strategy of UE_n obtained from a centralized algorithm.

Theorem 6. *In Hi-EECC, for the EECC game, PoA satisfies*

$$0 \leq \mathrm{PoA} \leq \frac{\sum_{n=1}^{|\mathcal{N}|} \min\{C_{n,l}, \tilde{C}^{min}_{n,m}, \hat{C}^{min}_n\}}{\sum_{n=1}^{|\mathcal{N}|} \max\{C_{n,l}, \tilde{C}^{max}_{n,m}, \hat{C}^{max}_n\}} \leq 1, \tag{57}$$

where

$$\tilde{C}^{max}_{n,m} = \frac{\sum_{n=1}^{|\mathcal{N}|} \gamma_n}{\gamma_n}\left(\phi_n\left(\frac{\omega_n}{\tilde{f}_m} + \frac{\delta_n}{\tilde{r}_m}\right) + (1-\phi_n)\frac{\delta_n \sigma'_n}{\tilde{r}_m}\right), \tag{58}$$

$$\hat{C}^{max}_n = \frac{\sum_{n=1}^{|\mathcal{N}|} \gamma_n \delta_n}{\gamma_n \tilde{r}_m}\left(\phi_n + (1-\phi_n)\sigma'_n\right) + \phi_n\left(\frac{\omega_n}{\hat{f}} + \frac{\delta_n}{\hat{r}}\right), \tag{59}$$

$$\tilde{C}^{min}_{n,m} = \phi_n\left(\frac{\omega_n}{\tilde{f}_m} + \frac{\delta_n}{\tilde{r}_m}\right) + (1-\phi_n)\frac{\delta_n \sigma'_n}{\tilde{r}_m}, \tag{60}$$

and

$$\hat{C}^{min}_n = \phi_n\left(\frac{\omega_n}{\hat{f}} + \frac{\delta_n}{\tilde{r}_m} + \frac{\delta_n}{\hat{r}}\right) + (1-\phi_n)\frac{\delta_n \sigma'_n}{\tilde{r}_m}. \tag{61}$$

Proof. Since $\overline{\boldsymbol{\lambda}}_{\boldsymbol{n}}$ is the optimal offloading strategy and $\boldsymbol{\lambda}^*_{\boldsymbol{n}}$ is one Nash equilibrium of the game, it is easily found that $0 \leq \mathrm{PoA} \leq 1$.

For UE_n, the resources allocated by ES_m are satisfied as follows

$$\frac{\gamma_n}{\sum_{n=1}^{|\mathcal{N}|} \gamma_n}\tilde{f}_m \leq f_{n,m} \leq \frac{\gamma_n}{\tilde{f}_m}, \tag{62}$$

$$\frac{\gamma_n}{\sum_{n=1}^{|\mathcal{N}|} \gamma_n}\tilde{r}_m \leq r_{n,m} \leq \frac{\gamma_n}{\tilde{r}_m}. \tag{63}$$

Let $f^{min}_{n,m} = \gamma_n \tilde{f}_m / \sum_{n=1}^{|\mathcal{N}|} \gamma_n$, $f^{max}_{n,m} = \tilde{f}_m$, $r^{min}_{n,m} = \gamma_n \tilde{r}_m / \sum_{n=1}^{|\mathcal{N}|} \gamma_n$, and $r^{max}_{n,m} = \tilde{r}_m$. Based on Equations (6), (9) and (18), for UE_n offloading its task to ES_m, the cost of the UE is satisfied, i.e.,

$$\begin{aligned}\tilde{C}_{n,m} &\leq \phi_n\left(\frac{\omega_n}{f^{min}_{n,m}} + \frac{\delta_n}{r^{min}_{n,m}}\right) + (1-\phi_n)\frac{\sigma'_n \delta_n}{r^{min}_{n,m}} \\ &= \frac{\sum_{n=1}^{|\mathcal{N}|} \gamma_n}{\gamma_n}\left(\phi_n\left(\frac{\omega_n}{\tilde{f}_m} + \frac{\delta_n}{\tilde{r}_m}\right) + (1-\phi_n)\frac{\delta_n \sigma'_n}{\tilde{r}_m}\right) \\ &= \tilde{C}^{max}_{n,m},\end{aligned} \tag{64}$$

and

$$\begin{aligned}\tilde{C}_{n,m} &\geq \phi_n\left(\frac{\omega_n}{f^{max}_{n,m}} + \frac{\delta_n}{r^{max}_{n,m}}\right) + (1-\phi_n)\frac{\sigma'_n \delta_n}{r^{max}_{n,m}} \\ &= \phi_n\left(\frac{\omega_n}{\tilde{f}_m} + \frac{\delta_n}{\tilde{r}_m}\right) + (1-\phi_n)\frac{\delta_n \sigma'_n}{\tilde{r}_m} \\ &= \tilde{C}^{min}_{n,m}.\end{aligned} \tag{65}$$

For UE_n offloading its task to the cloud, the cost of the UE is satisfied, i.e.,

$$\begin{aligned}\hat{C}_n &\leq \phi_n\left(\frac{\omega_n}{\hat{f}} + \frac{\delta_n}{r^{min}_{n,m}} + \frac{\delta_n}{\hat{r}}\right) + (1-\phi_n)\frac{\sigma'_n \delta_n}{r^{min}_{n,m}} \\ &= \phi_n\left(\frac{\omega_n}{\hat{f}} + \frac{\delta_n}{\hat{r}}\right) + \frac{\sum_n^{|\mathcal{N}|} \gamma_n \delta_n}{\gamma_n \tilde{r}_m}\left(\phi_n + (1-\phi_n)\sigma'_n\right) \\ &= \hat{C}^{max}_n,\end{aligned} \tag{66}$$

and

$$\begin{aligned}\hat{C}_n &\geq \phi_n\left(\frac{\omega_n}{\hat{f}} + \frac{\delta_n}{r^{max}_{n,m}} + \frac{\delta_n}{\hat{r}}\right) + (1-\phi_n)\frac{\sigma'_n \delta_n}{r^{max}_{n,m}} \\ &= \phi_n\left(\frac{\omega_n}{\hat{f}} + \frac{\delta_n}{\tilde{r}_m} + \frac{\delta_n}{\hat{r}}\right) + \frac{(1-\phi_n)\delta_n \sigma'_n}{\tilde{r}_m} \\ &= \hat{C}^{min}_n.\end{aligned} \tag{67}$$

For UE_n executing its task locally, the cost of the UE is certain, i.e., $C_{n,l} = \phi_n \omega_n / f_n + (1-\phi_n)\sigma_n \omega_n / f_n$. Based on the above, we obtain

$$\sum_{n=1}^{|\mathcal{N}|} C_n(\overline{\boldsymbol{\lambda}}_{\boldsymbol{n}}) \geq \sum_{n}^{|\mathcal{N}|} \min\{C_{n,l}, \tilde{C}^{min}_{n,m}, \hat{C}^{min}_n\}, \tag{68}$$

and

$$\sum_{n=1}^{|\mathcal{N}|} C_n(\boldsymbol{\lambda}^*_{\boldsymbol{n}}) \leq \sum_{n}^{|\mathcal{N}|} \max\{C_{n,l}, \tilde{C}^{max}_{n,m}, \hat{C}^{max}_n\}. \tag{69}$$

Therefore, we have

$$\mathrm{PoA} \leq \frac{\sum_{n=1}^{|\mathcal{N}|} \min\{C_{n,l}, \tilde{C}^{min}_{n,m}, \hat{C}^{min}_n\}}{\sum_{n=1}^{|\mathcal{N}|} \max\{C_{n,l}, \tilde{C}^{max}_{n,m}, \hat{C}^{max}_n\}}. \tag{70}$$

Thus, we have the conclusion. □

In Ho-EECC, PoA is defined as

$$\mathrm{PoA} = \frac{\sum_{n=1}^{|\mathcal{N}|} C'_n(\overline{\boldsymbol{\lambda}}'_{\boldsymbol{n}})}{\max_{\boldsymbol{\lambda}_{\boldsymbol{n}}} \sum_{n=1}^{|\mathcal{N}|} C'_n(\boldsymbol{\lambda}^*_{\boldsymbol{n}})}, \tag{71}$$

where $\overline{\lambda}'_n$ is an optimal offloading strategy of UE_n obtained from a centralized algorithm.

Theorem 7. *In Ho-EECC, for the EECC game, PoA satisfies*

$$0 \leq \text{PoA} \leq \frac{\sum_{n=1}^{|\mathcal{N}|} \min\{C_{n,l}, \tilde{C}^{min}_{n,m}, \hat{C}'_n\}}{\sum_{n=1}^{|\mathcal{N}|} \max\{C_{n,l}, \tilde{C}^{max}_{n,m}, \hat{C}'_n\}} \leq 1. \tag{72}$$

Proof. Based on Equations (16), (17), and (19), in Ho-EECC, the cost of a_n executed in the cloud is

$$\hat{C}'_n = \phi_n \left(\frac{\omega_n}{\hat{f}} + \frac{\delta_n}{\hat{r}'_1} + \frac{\delta_n}{\hat{r}'_2} \right) + \sigma'_n (1 - \phi_n) \left(\frac{\delta_n}{\hat{r}'_1} + \frac{\delta_n}{\hat{r}'_2} \right). \tag{73}$$

As shown in the proof of Theorem 6, we know that the minimum cost of UE_n responded to by ES_m is $\tilde{C}^{min}_{n,m}$. Moreover, the maximum cost of a_n executed by ES_m is $\tilde{C}^{max}_{n,m}$. Therefore, we have

$$\sum_n^{|\mathcal{N}|} C'_n(\overline{\boldsymbol{\lambda}}'_n) \geq \sum_n^{|\mathcal{N}|} \min\{C_{n,l}, \tilde{C}^{min}_{n,m}, \hat{C}'_n\}, \tag{74}$$

and

$$\sum_n^{|\mathcal{N}|} C'_n(\boldsymbol{\lambda}^*_n) \leq \sum_n^{|\mathcal{N}|} \max\{C_{n,l}, \tilde{C}^{max}_{n,m}, \hat{C}'_n\}. \tag{75}$$

Based on the above inequalities, we obtain

$$0 \leq \text{PoA} \leq \frac{\sum_n^{|\mathcal{N}|} \min\{C_{n,l}, \tilde{C}^{min}_{n,m}, \hat{C}'_n\}}{\sum_n^{|\mathcal{N}|} \max\{C_{n,l}, \tilde{C}^{max}_{n,m}, \hat{C}'_n\}} \leq 1.$$

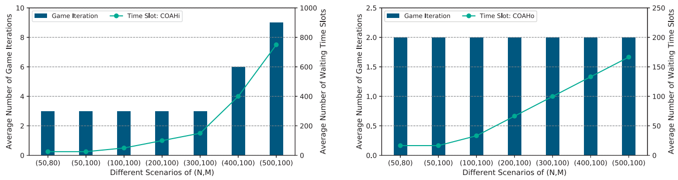
Thus, we have the conclusion. □

7 Experimental Evaluation

In this section, extensive experiments with real-world data are conducted to demonstrate the convergence and performance of the proposed algorithms. The comparison between the developed algorithms (i.e., COAHi and COAHo) is actually the comparison between Hi-EECC and Ho-EECC. The scalability and applicability of Hi-EECC and Ho-EECC under the influence of various factors are also comprehensively studied. We present three important conclusions for choosing specific computing architectures in the different application scenario through the experimental analysis.

7.1 Parameter Configuration

In the experiments, we assume that there are six service areas, i.e., $|\mathcal{S}| = 6$. UEs and ESs are randomly located in one of the service areas. Different numbers of UEs and ESs are generated to evaluate the proposed algorithms. Most parameters used in the experiments are real-world values obtained from other work. Specifically, the computing capacity of UE_n is randomly taken from $\{0.5, 0.8, 1\}$ GHZ [29], [36]. The computing power of ES_m is randomly assigned from $\{5, 6, 8, 9\}$ GHz [37]. Furthermore, the computing resource of the cloud is $\hat{f} = 10$ GHz [29]. The communication resource of ES_m is $\tilde{r}_m = 9.97R$ Mbps [38], where $R \in [5, 10]$ is a random integer variable and reflects the heterogeneity of different ESs. The communication resource of



(a) (b)

Fig. 4. Average number of game rounds for obtaining a Nash equilibrium, and average number of waiting time slots needed by UEs to determine an offloading strategy in two computing architectures. (a) Hi-EECC. (b) Ho-EECC.

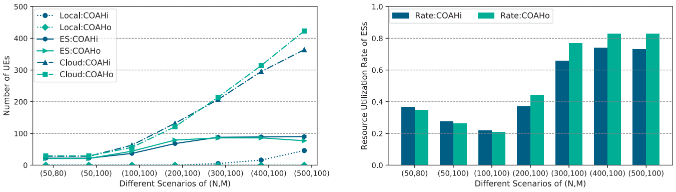
the cloud is $\hat{r} = 99.7$ Mbps. In Ho-EECC, without loss of generality, let $\hat{r}'_1 = \hat{r}'_2 = 1.52$ Mbps [27]. In addition, σ_n, σ'_n, and ϕ_n are randomly taken from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. γ_n is randomly assigned from $\{1, 2, 3, 4, 5\}$.

To reflect the heterogeneity of UEs, we assume that UEs can execute three kinds of tasks: facial recognition [39], video game [40], and video transcoding [41]. Since it is difficult for us to directly obtain the workload (i.e., ω_n) of a task, we introduce the processing density (represented by ν_n), which is quantified by the number of cycles per bit [27]. The number of CPU cycles required to complete a task can be calculated through $\omega_n = \delta_n \nu_n$ [27]. The processing densities of the above tasks use the real-world measurement data, i.e., facial recognition: 2339 cycles/bit [39]; video game : 2640 cycles/bit [40]; and video transcoding: 1000 cycles/bit [41]. Moreover, the data size of a task is randomly assigned from $\{1, 2, 3, 4, 5\}$ MB.

7.2 Experimental Results and Analysis

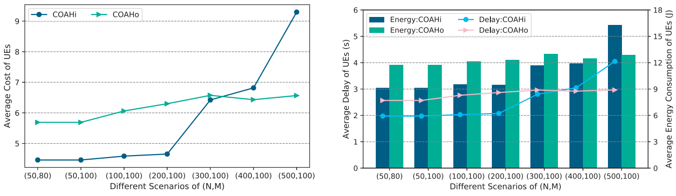
7.2.1 The convergence of algorithms

Fig. 4 shows the average number of game rounds required for Algorithm 1 to find a Nash equilibrium, and the average number of waiting time slots needed for COAHi and COAHo to determine an offloading strategy in the two computing architectures. During the iteration process, the algorithms allow only one UE to update its strategy at a time, while other UEs are in a waiting state. In this paper, the time required for UE_n to determine an offloading strategy is represented by the number of waiting time slots. In reality, a time slot is very short and at the time scale of microseconds [29]. Therefore, as shown in Figs. 4a and 4b, as the number of UEs (i.e., $N = |\mathcal{N}|$) increases, the average number of waiting time slots increases. As shown in Fig. 4a, in Hi-EECC, because the resources that UEs can obtain from ESs are very limited, as the number of UEs increases, the competition between UEs intensifies, so the number of game rounds and waiting time slots increases rapidly. However, as shown in Fig. 4b, UEs can obtain sufficient resources from the cloud to meet their own demands in Ho-EECC. Therefore, the number of game rounds does not change with the increase in the number of UEs. Compared with Hi-EECC, this is the reason why the average number of waiting time slots in Ho-EECC is less. The explanation is proved again by Fig. 5a. We can also know from the experiment that the change in the number of ESs (i.e., $M = |\mathcal{M}|$) does not affect the convergence of the algorithms, which is consistent with Theorems 4 and 5.



(a) (b)

Fig. 5. Comparison between COAHi and COAHo in the number of tasks executed by different entities (a), and resource utilization rate of ESs (b).



(a) (b)

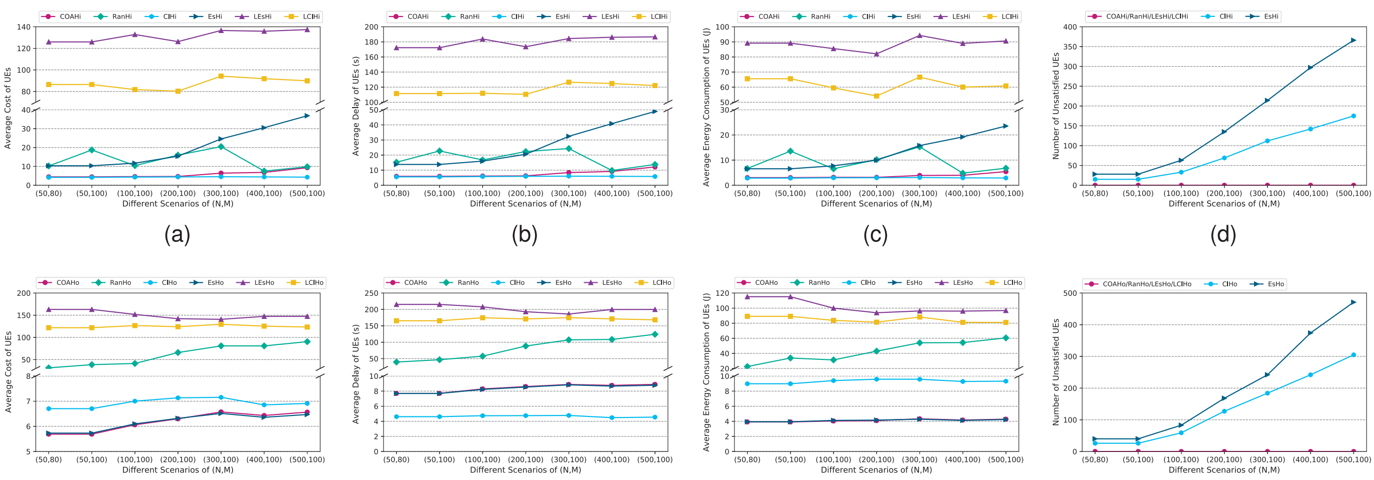
Fig. 6. Comparison between COAHi and COAHo in terms of cost (a), and delay and energy consumption (b).

7.2.2 The impact of N and M

Fig. 5a shows the number of UEs that execute applications locally (i.e., Local: COAHi and Local: COAHo), upload tasks to the ESs (i.e., ES: COAHi and ES: COAHo), and responded by the cloud (i.e., Cloud: COAHi and Cloud: COAHo) in the two computing architectures. The figure shows that as the number of UEs increases, increasingly more UEs submit their tasks to the cloud. In addition, in Hi-EECC, although ESs are trying their best to satisfy more UEs, increasingly more UEs still choose to perform tasks locally. Moreover, in Hi-EECC and Ho-EECC, since the resources of ESs are limited, as the resources of ESs are exhausted, the number of requests that the ESs can respond to reaches the upper limit. A comparison of the resource utilization rate of ESs between the two computing architectures is depicted in Fig. 5b, i.e., Rate: COAHi and Rate: COAHo. The resource utilization rate of ESs refers to the ratio of the number of ESs responding to UEs' requests to the total number of ESs, i.e., $\sum_{m=1}^{|\mathcal{M}|}\mathbb{I}\{\sum_{n=1}^{|\mathcal{N}|}\tilde{\lambda}_{n,m}\geq 1\}/|\mathcal{M}|$, where $\mathbb{I}\{\cdot\}=\{0,1\}$ is an indicator function. $\mathbb{I}\{\cdot\}=1$ when the input parameter of the function is true. Otherwise, $\mathbb{I}\{\cdot\}=0$. The resource utilization rate of ESs can help select the appropriate computing architecture in the different application scenario, thereby reducing the overhead required to maintain the ESs running. As shown in Fig. 5b, COAHi performs better when $N\leq 100$, and COAHo performs better when $N\geq 200$. Moreover, as shown in Figs. 6a and 6b, COAHi performs better when $N\leq 300$, and COAHo performs better when $N\geq 500$. It can be seen from the above figures that low-latency data transmission offsets the resource shortcomings of ESs. However, the resources of ESs are unable to cope with the large-scale user scenario. Based on the above discussions, we can reach our first important conclusion. In terms of cost, delay, and energy consumption of UEs, as well as the resource utilization rate of ESs, Hi-EECC is more suitable for the small-scale user scenario, while Ho-EECC is more suitable for the large-scale user scenario.

7.2.3 The performance of algorithms

To evaluate the performance of COAHi, we use the following five schemes as the baselines. (1) RanHi: UEs randomly determine an offloading decision. (2) ClHi: All UEs' tasks are executed by the cloud. (3) EsHi: All UEs request ESs to execute their tasks. (4) LEsHi: UEs determine the offloading decision between itself and ESs. (5) LClHi: All UEs request ESs to process their tasks. The deadline unsatisfied tasks are further uploaded to the cloud by the ESs. The difference between EsHi and LClHi is in whether to upload the deadline unsatisfied tasks to the cloud. In Ho-EECC, we use the same baselines. However, the differences are that UEs must consider their deadline when making a strategy, and UEs can directly request the cloud. To distinguish the two computing architectures, the relevant benchmarks are named RanHo, ClHo, EsHo, LEsHo, and LClHo. It should be noted that when all UEs can determine offloading decisions only between itself and ESs, both LEsHi and LEsHo determine the decisions by using Algorithm 1. Since the Nash equilibrium is not unique, the strategies of LEsHi and LEsHo have some random differences in terms of cost, delay, and energy consumption of UEs. Moreover, as mentioned above, the comparison between the



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 7. Comparison between the proposed algorithms and baselines in terms of cost, delay, energy consumption, and the number of deadline unsatisfied UEs in two computing architectures. (a)-(d) Hi-EECC. (e)-(h) Ho-EECC.

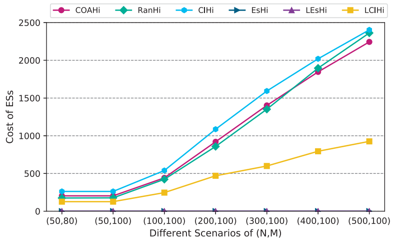


Fig. 8. Comparison of the cost of ESs between different algorithms.

developed algorithms and the baselines is actually the comparison between computing architectures such as Hi-EECC, Ho-EECC, EC, CC, and edge-cloud computing.

Fig. 7 compares the performance of the algorithms in the different scenario in detail. As shown in Fig. 7, although algorithms EsHi and ClHo perform better in terms of cost, delay, and energy consumption, many UEs' QoS demands cannot be satisfied. The cost of ESs using different schemes is shown in Fig. 8. It can be seen from the figures that COAHi and COAHo perform better than the baselines. Thus, through the above experiments, we find that the proposed algorithms perform better in terms of the cost, delay, energy consumption, and QoS demand of UEs. Moreover, compared with EC and CC, EECC shows unique advantages. UEs can handle some real-time tasks based on their own resources. ESs can provide UEs with low-latency and low energy consumption services for latency-sensitive tasks. The cloud can provide services for UEs to process their computation-intensive tasks. The end, edge, and cloud are complementary to one another and can more flexibly adapt to various user requirements.

7.2.4 The impact of application type

As shown in Equations (3), (6), and (7), we know that ω_n affects the computation delay of tasks. As shown in Equations (9), (10), and (12), we know that δ_n affects the communication delay of tasks. Based on the data size and workload, we classify the application into communication-intensive tasks and computation-intensive tasks. To better reflect the effectiveness of the algorithms, and evaluate the adaptability of two computing architectures to different application types, we introduce two multipliers α and β to increase the data size and workload of tasks, respectively. As shown in Figs. 9a, 9b, and 9c, if the data size is expanded to α times the original data size, the performance of COAHi gradually becomes better than that of COAHo. As mentioned above, Ho-EECC is more suitable for the large-scale user scenario. When the user scale is fixed, the increase in the data size directly prolongs the transmission delay of applications. The long distance and low transmission rate between UEs and the cloud weaken the advantage of Ho-EECC. As shown in Fig. 9d, the UEs that originally initiated requests to the cloud began to request ESs to perform their tasks. Based on the above discussions, we can reach our second important conclusion. For communication-intensive tasks, the cost of UEs in Hi-EECC is less than the cost of UEs in Ho-EECC. That is, whether in the large-scale user scenario or small-scale user scenario, we can conclude that Hi-EECC is a better choice for communication-intensive tasks. The reason for this phenomenon is that the communication delay dominates the cost of UEs. Obviously, uploading tasks to ESs closer to UEs is more in line with the UEs' demands.

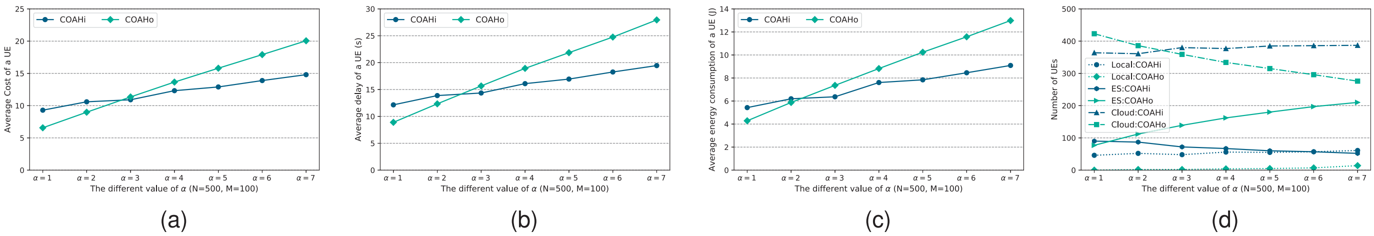


Fig. 9. Comparison between COAHi and COAHo in terms of cost (a), delay (b), energy consumption (c), and the number of UEs responded to by different entities (d) when α takes different values.

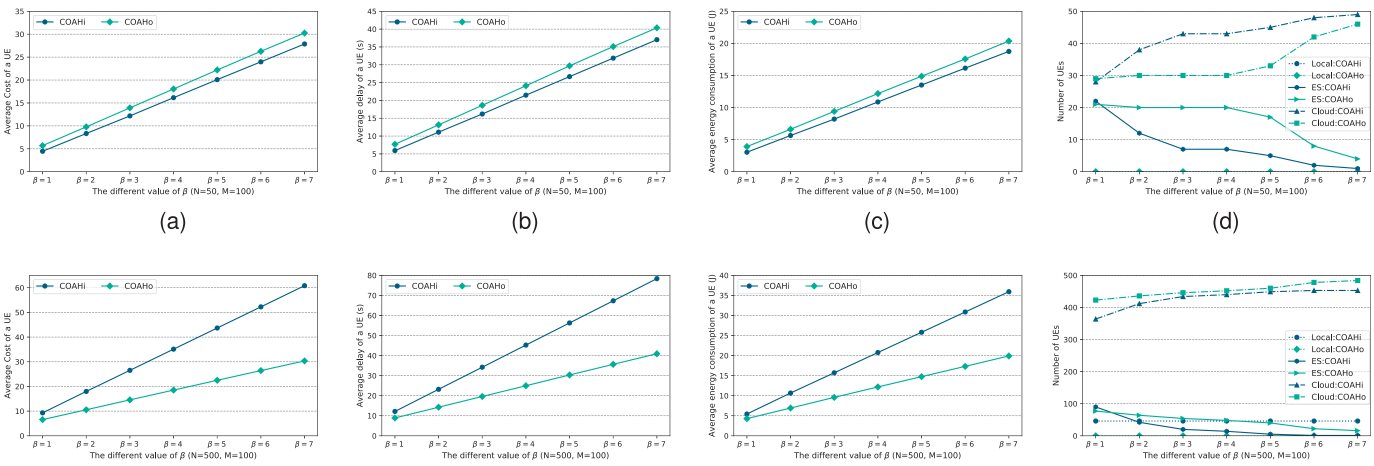


Fig. 10. Comparison between COAHi and COAHo in terms of cost (a), delay (b), energy consumption (c), and the number of UEs responded to by different entities (d) when β takes different values.

As shown in Figs. 10a, 10b, and 10c, as the workload is expanded to β times the original workload, we can see that the performance of COAHi is better than that of COAHo in the small-scale user scenario. However, as shown in Figs. 10e, 10f, and 10g, it can be seen that the performance of COAHo is better than that of COAHi in the large-scale user scenario. Based on the above discussions, we can reach our third important conclusion. For computation-intensive tasks, we can conclude that Hi-EECC is a better choice for UEs in the small-scale user scenario, and Ho-EECC is a better choice for UEs in the large-scale user scenario. As shown in Figs. 10d and 10h, with an increase in workload, UEs tend to initiate requests to the cloud regardless of the computing architectures. The reason for this phenomenon is that computation delay dominates the cost of UEs. The resources of ESs are unable to meet the demands of large-scale users. Hence, we can again confirm that EECC can improve the resource utilization of UEs, ESs, and the cloud to better serve users with different demands.

8 Conclusion and Future Work

In this paper, we construct a potential game for the EECC environment, in which each UE selfishly minimizes its payoff, and investigate the computation offloading strategy optimization for UEs in Hi-EECC and Ho-EECC. Accordingly, we develop two potential game-based algorithms, i.e., COAHi and COAHo, to determine the best offloading strategies for all UEs. The scalability and applicability of Hi-EECC and Ho-EECC under the influence of various factors are also comprehensively studied. We present three important conclusions for choosing specific computing architectures in the different application scenario through the experimental analysis. The main conclusions are as follows: (1) In terms of cost, delay, and energy consumption of UEs, as well as the resource utilization rate of ESs, Hi-EECC is more suitable for the small-scale user scenario, while Ho-EECC is more suitable for the large-scale user scenario. (2) For communication-intensive tasks, whether in the large-scale or small-scale user scenario, the cost of UEs in Hi-EECC is lower than the cost of UEs in Ho-EECC; that is, Hi-EECC is a better choice for UEs. (3) For computation-intensive tasks, Hi-EECC is a better choice for UEs in the small-scale user scenario, and Ho-EECC is a better choice for UEs in the large-scale user scenario.

The assumption that ESs in the same service area are homogeneous is a limitation of this paper, which is an issue left for our future research. Moreover, the paper opens many research topics in EECC. In our future work, we will first study the impact of mobility on the cost of UEs and the service mode of EECC. The combination of Hi-EECC and Ho-EECC is also an interesting direction worthy of investigation.

Acknowledgments

We would like to express our gratitude to the associate editor and anonymous reviewers for their comments which are very important to improve the quality of the manuscript.

References

[1] C. Liu *et al.*, "Coexe: An efficient co-execution architecture for real-time neural network services," in *Proc. 57th ACM/IEEE Des. Automation Conf.*, 2020, pp. 1–6.

[2] C. Qiu, X. Wang, H. Yao, J. Du, F. R. Yu, and S. Guo, "Networking integrated cloud-edge-end in IoT: A blockchain-assisted collective Q-learning approach," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 12694–12704, Aug. 2021.

[3] B. Yang, X. Cao, J. Bassey, X. Li, and L. Qian, "Computation offloading in multi-access edge computing: A multi-task learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 9, pp. 2745–2762, Sep. 2021.

[4] W. A. Simon, Y. M. Qureshi, M. Rios, A. Levisse, M. Zapater, and D. Atienza, "BLADE: An in-cache computing architecture for edge devices," *IEEE Trans. Comput.*, vol. 69, no. 9, pp. 1349–1363, Sep. 2020.

[5] L. Chen, C. Shen, P. Zhou, and J. Xu, "Collaborative service placement for edge computing in dense small cell networks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 377–390, Feb. 2021.

[6] Y. Ma, W. Liang, J. Li, X. Jia, and S. Guo, "Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks," *IEEE Trans. Mob. Comput.*, to be published, doi: 10.1109/TMC.2020.3006507.

[7] J. Cui, L. Wei, H. Zhong, J. Zhang, Y. Xu, and L. Liu, "Edge computing in vanets-an efficient and privacy-preserving cooperative downloading scheme," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1191–1204, Jun. 2020.

[8] J. Hu, K. Li, C. Liu, J. Chen, and K. Li, "Coalition formation for deadline-constrained resource procurement in cloud computing," *J. Parallel Distrib. Comput.*, vol. 149, pp. 1–12, 2021.

[9] G. Liu, Z. Xiao, G. Tan, K. Li, and A. T. Chronopoulos, "Game theory-based optimization of distributed idle computing resources in cloud environments," *Theor. Comput. Sci.*, vol. 806, pp. 468–488, 2020.

[10] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *Proc. 39th IEEE Conf. Comput. Commun.*, 2020, pp. 257–266.

[11] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019.

[12] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical fog-cloud computing for IoT systems: A computation offloading game," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3246–3257, Aug. 2018.

[13] M. Du, Y. Wang, K. Ye, and C. Xu, "Algorithmics of cost-driven computation offloading in the edge-cloud environment," *IEEE Trans. Comput.*, vol. 69, no. 10, pp. 1519–1532, Oct. 2020.

[14] R. Fantacci and B. Picano, "Performance analysis of a delay constrained data offloading scheme in an integrated cloud-fog-edge computing system," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12 004–12 014, Oct. 2020.

[15] T. Wang, D. Zhao, S. Cai, W. Jia, and A. Liu, "Bidirectional prediction-based underwater data collection protocol for end-edge-cloud orchestrated system," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4791–4799, Jul. 2020.

[16] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[17] M. Ejaz, T. Kumar, M. Ylianttila, and E. Harjula, "Performance and efficiency optimization of multi-layer IoT edge architecture," in *Proc. 2nd 6G Wireless Summit*, 2020, pp. 1–5.

[18] Y. Wu, K. Ni, C. Zhang, L. P. Qian, and D. H. K. Tsang, "Noma-assisted multi-access mobile edge computing: A joint optimization of computation offloading and time allocation," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12 244–12 258, Dec. 2018.

[19] C. You, K. Huang, H. Chae, and B. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wirel. Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[20] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1619–1632, Aug. 2018.

[21] J. Hu, K. Li, C. Liu, and K. Li, "Game-based task offloading of multiple mobile devices with QoS in mobile edge computing systems of limited computation capacity," *ACM Trans. Embed. Comput. Syst.*, vol. 19, no. 4, pp. 29:1–29:21, 2020.

[22] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial iot-edge-cloud computing environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 12, pp. 2759–2774, Dec. 2019.

[23] K. Peng, H. Hualong, S. Wan, and V. Leung, "End-edge-cloud collaborative computation offloading for multiple mobile users in heterogeneous edge-server environment," *Wireless Netw.*, pp. 1–12, 2020 [Online]. Available: https://doi.org/10.1007/s11276-020-02385-1.

[24] C. Sun *et al.*, "Task offloading for end-edge-cloud orchestrated computing in mobile networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2020, pp. 1–6.
[25] P. Cong, G. Xu, T. Wei, and K. Li, "A survey of profit optimization techniques for cloud providers," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 26:1–26:35, 2020.
[26] Q. He *et al.*, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, Mar. 2020.
[27] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510–2523, Dec. 2015.
[28] J. L. D. Neto, S. Yu, D. F. Macedo, J. M. S. Nogueira, R. Langar, and S. Secci, "ULOOF: A user level online offloading framework for mobile edge computing," *IEEE Trans. Mob. Comput.*, vol. 17, no. 11, pp. 2660–2674, Nov. 2018.
[29] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.
[30] W. Zhang, Y. Wen, K. Guan, D. C. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wirel. Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.
[31] M. Hu, Z. Xie, D. Wu, Y. Zhou, X. Chen, and L. Xiao, "Heterogeneous edge offloading with incomplete information: A minority game approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 9, pp. 2139–2154, Sep. 2020.
[32] D. Monderer and L. S. Shapley, "Potential games," *Games Economic Behav.*, vol. 14, no. 1, pp. 124–143, 1996.
[33] J. R. Marden, G. Arslan, and J. S. Shamma, "Cooperative control and potential games," *IEEE Trans. Syst. Man, Cybern. B*, vol. 39, no. 6, pp. 1393–1407, Dec. 2009.
[34] Q. He *et al.*, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, Mar. 2020.
[35] T. Roughgarden, *Selfish Routing and the Price of Anarchy*. Cambridge, MA, USA: MIT Press, 2005.
[36] O. Muñoz, A. Pascual-Iserte , and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.
[37] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. B. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proc. IEEE Symp. Comput. Commun.*, 2012, pp. 59–66.
[38] J. McChesney, N. Wang, A. Tanwer, E. de Lara, and B. Varghese, "DeFog: Fog computing benchmarks," in *Proc. 4th ACM/IEEE Symp. Edge Comput.*, S. Chen, R. Onishi, G. Ananthanarayanan, and Q. Li, Eds., 2019, pp. 47–58.
[39] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, 2012, pp. 945–953.
[40] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proc. 8th ACM Int. Conf. Mobile Syst. Appl. Services*, 2010, pp. 49–62.
[41] J. Kwak, O. Choi, S. Chong, and P. Mohapatra, "Processor-network speed scaling for energy-delay tradeoff in smartphone applications," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1647–1660, Jun. 2016.

Yan Ding (Student Member, IEEE) received the BS degree in software engineering from the North University of China, in 2014, and the MS degree in computer application technology from Xinjiang University, Urumqi, China, in 2018. He is currently working toward the PhD degree at the Hunan University, China. His research interests include mobile edge computing, data analysis, machine learning, and network security. He has published four papers in journals and conference, including *IEEE Transactions on Industrial Informatics*, *Journal of Parallel and Distributed Computing*, *Computers & Security*, and the 17th IEEE International Symposium on Parallel and Distributed Processing with Applications (IEEE ISPA 2019). He obtained the Outstanding Paper Award in IEEE ISPA 2019. He is a student member of CCF.

Kenli Li (Senior Member, IEEE) received the PhD degree in computer science from the Huazhong University of Science and Technology, China, in 2003. He was a visiting scholar with the University of Illinois at Urbana-Champaign from 2004 to 2005. He is currently a full professor of computer science and technology with Hunan University, the dean of the College of Information Sciences and Engineering, Hunan University, and the director in the National Supercomputing Center in Changsha. His major research interests include parallel computing, high-performance computing, and grid and cloud computing. He has published more than 160 research papers in international conferences and journals such as *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *Journal of Parallel and Distributed Computing*, ICPP, ICDCS, etc. He serves on the editorial board of the *IEEE Transactions on Computers*. He is an outstanding member of CCF.

Chubo Liu (Member, IEEE) received the BS and PhD degrees in computer science and technology from Hunan University, China, in 2011 and 2016, respectively. He is currently an associate professor of computer science and technology with Hunan University. His research interests include game theory, approximation and randomized algorithms, cloud and edge computing. He has published over 20 papers in journals and conferences such as the *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Industrial Informatics*, *IEEE Internet of Things Journal*, *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, *Theoretical Computer Science*, ICPADS, HPCC, and NPC. He won the Best Paper Award in IFIP NPC 2019 and the IEEE TCSC Early Career Researcher (ECR) Award in 2019. He is a member of CCF.

Keqin Li (Fellow, IEEE) is a SUNY distinguished professor of computer science with the State University of New York. He is also a national distinguished professor with Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber-physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has authored or coauthored nearly 800 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He holds more than 60 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top 10 most influential scientists in distributed computing based on a composite indicator of Scopus citation database. He has chaired many international conferences. He is currently an associate editor of the ACM Computing Surveys and CCF Transactions on High Performance Computing. He has served on the editorial boards of *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Services Computing*, and *IEEE Transactions on Sustainable Computing*.