Joint Optimization of Computing Offloading and Service Caching in Edge Computing-Based Smart Grid

Huan Zhou, *Member, IEEE*, Zhenyu Zhang, *Student Member, IEEE*, Dawei Li, *Member, IEEE*, and Zhou Su, *Senior Member, IEEE*

Abstract—With the continuous expansion of the power Internet of Things (IoT) and the rapid increase in the number of Smart Devices (SDs), the data generated by SDs has exponentially increased. The traditional cloud-based smart grid cannot meet the low latency and high reliability requirements of emerging applications. By moving computing, data, and services from the centralized cloud to Edge Servers (ESs), edge computing exhibits excellent performance in communication delay and traffic reduction. Simultaneously, service caching also shows attractive advantages in handling the surge in data traffic. In this paper, we consider the joint optimization of computing offloading and service caching in edge computing-based smart grid, and formulate the problem as a Mixed-Integer Non-Linear Program (MINLP), aiming to minimize the task cost of the system. The original problem is decomposed into an equivalent master problem and sub-problem, and a Collaborative Computing Offloading and Resource Allocation Method (CCORAM) is proposed to solve the optimization problem, which includes two low-complexity algorithms. Specifically, a gradient descent allocation algorithm is first proposed to determine the computing resource allocation strategy, and then a game theory-based algorithm is proposed to determine the computing strategy. Simulation results show that CCORAM with low time complexity is very close to the optimal method, and performs much better than other benchmark methods.

Index Terms—Computing offloading, edge computing, game theory, service caching, smart grid

1 Introduction

With the progress of technology and the expansion in the power industry, the traditional power is evolving into a more powerful data-driven intelligent era [1]. The traditional power grid has exposed some problems, such as the inability to deploy intelligently to meet the electricity demand of consumers, and the failure to process data in real time. Accordingly, the cloud computing-based smart grid is considered to be an attractive emerging technology trend to solve the above problems in recent years [2]–[4]. Different segments of smart grid applications (i.e., data aggregation, data processing, data analysis, etc.) often have very high requirements for processing latency, transmission bandwidth and data privacy. The cloud computing-based smart grid provides higher computing power and data storage capacity to handle the increasing data traffic and tasks of SDs [5]. Although the current cloud computing-based smart grid can solve some of these problems, it will cause huge communication delay and consume a large amount of network bandwidth, which cannot meet the requirements of low latency and real-time processing. Specifically, if a large number of tasks pour into the cloud, it will aggravate the communication pressure of the core network [6], [7].

Recently, the concept of edge computing has been proposed as a supplement to cloud computing, which has aroused great interest in academia and industry [8]–[11]. Specifically, by moving computing, data, and services from the centralized cloud to ESs, edge computing exhibits excellent performance in communication delay and traffic reduction. Therefore, edge computing as a new way to solve the above problems is introduced into smart grid to relieve the pressure of cloud server [12], [13]. ESs are placed at the edge of the network, which are responsible for a limited number of SDs. In edge computing-based smart grid, SDs can offload all or part of the computing tasks to ESs through wireless channels, thereby speeding up task processing and reducing delay and energy consumption of the system. Meanwhile, it is no longer necessary to transfer data on remote network, and the security and stability of the smart grid are controllable.

Although computing offloading has become the research focus in edge computing, the heterogeneity of computing tasks and service caching is often ignored. Specifically, service caching means that the ESs cache application services

- *Huan Zhou and Zhenyu Zhang are with the College of Computer and Information Technology, Hubei Key Laboratory of Intelligent Vision Based Monitoring for Hydroelectric Engineering, China Three Gorges University, Yichang 443002, China. E-mail: zhouhuan117@gmail.com, zyzhang0731@163.com.*
- *Dawei Li is with the Department of Computer Science, Montclair State University, Montclair, NJ 07041 USA. E-mail: dawei.li@montclair.edu.*
- *Zhou Su is with the School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an 710000, China. E-mail: zhousu@xjtu.edu.cn.*

Manuscript received 18 July 2021; revised 16 Dec. 2021; accepted 19 Jan. 2022. Date of publication 12 Apr. 2022; date of current version 7 June 2023.
This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62172255, 61872221, and U20A20175.
(Corresponding author: Huan Zhou.)
Recommended for acceptance by R. Deng, C.-W. Ten, C. Li, D. Niyato, and F. Teng.
Digital Object Identifier no. 10.1109/TCC.2022.3163750

and related databases, so that they can process corresponding computing tasks to reduce completion time and energy consumption [14], [15]. Service is an abstraction of applications that is hosted by the ES and requested by SDs. Taking intelligent monitoring in smart grid as an example, processing computing tasks requires not only sensor data, but also data from surrounding infrastructure to provide services. The sensor data is collected by SDs, including the appearance, operating status of power equipment in the substation, and abnormal conditions on the line. The surrounding infrastructure data is stored in the cloud server, which includes weather conditions, protection around the substation, etc. To make the task offloading efficient, the surrounding infrastructure data and sensor data have to be collocated with the computing task when it is executed by an ES. However, if the ES does not cache the required services (i.e., surrounding infrastructure data) when receiving the offloaded task, it will initiate a request to the cloud server and download it, thus increasing the latency. Furthermore, due to the limited caching capacity, each ES can only cache a small portion of services. Therefore, how to cache the required services is very important, and may significantly affect the edge computing performance. In this paper, we study the problem of joint optimization of computing offloading and service caching in edge computing-based smart grid. We first formulate the problem as a Mixed-Integer Non-Linear Program (MINLP), aiming to minimize the task cost of the system. We further modify the constraints of the original problem, and decompose the original problem into an equivalent master problem and sub-problem. Then, we propose two low-complexity algorithms to solve the optimization problem. Finally, the effectiveness of our proposed method is demonstrated by simulation. The main contributions of this paper are summarized as follows:

- We propose an edge computing-based smart grid architecture featuring computing offloading with service caching, and formulate the optimization problem of joint computing offloading and service caching as a MINLP, aiming to minimize the task cost of the system.
- In order to solve the optimization problem, we decompose the original problem into an equivalent master problem and sub-problem, and propose a Collaborative Computing Offloading and Resource Allocation Method (CCORAM), which includes two low-complexity algorithms. Specifically, a gradient descent allocation algorithm is proposed to determine the resource allocation strategy, and then a game theory-based algorithm is proposed to obtain the computing strategy.
- Numerical results show that CCORAM with low time complexity is very close to the optimal method, and greatly outperforms the other benchmark methods.

The remainder of this paper is organized as follows. Sections 2 and 3 describe the related work and the system model, respectively. Section 4 formulates the problem as a MINLP to minimize the total task cost of the system. Section 5 decomposes the original problem into an equivlent master problem and a sub-problem. Section 6 presents two low-complexity algorithms to solve the corresponding optimization problems. Furthermore, we analyze the simulation results in Section 7. Finally, the conclusion and future work are introduced in Section 8.

2 Related Work

In recent years, the application of edge computing in the smart grid has attracted significant attention and research, which effectively reduces the processing latency of computationally intensive tasks and protects data privacy [16]–[17]. At present, some studies have focused on building smart grid architecture with edge computing. The authors in [18] proposed an architecture for smart grid data analysis based on edge computing to achieve the accuracy of smart grid applications. In [19], a distributed anomaly detection architecture for smart grid based on edge computing is proposed, which improves the security of smart grid by detecting anomalies in power consumption data through the cooperation of distributed devices at the edge of the network. In [20], the authors suggested the adoption of edge computing for communication networks, and explored the impact of edge computing on the response time and transfer rate to handle a large amount of data generated in smart grid. In [21], the authors introduced a smart grid framework based on edge computing to achieve high-performance real-time monitoring, and a heuristic algorithm using simulated annealing strategy is proposed to solve task scheduling problem.

Furthermore, some studies focus on investigating the computing offloading and resource allocation in edge computing-based smart grid. In [22], the authors proposed a joint edge computing and device-to-device based computation offloading scheme in smart grid, and minimized the energy consumption while meeting the requested bandwidth and delay. In [23], the authors proposed an efficient and secure multidimensional data aggregation scheme for fog computing-based smart grid to reduce computing and communication costs, which also can resist various security attacks and preserve the user's privacy. The authors in [24] introduced edge computing into smart grid fault detection system, and proposed an optimal allocation method of communication and computing resources to maximize the throughput of the system. The authors in [25] studied the problem of resource scheduling in cloud edge-based smart grid, and proposed a new hybrid artificial bee ant colony optimization algorithm to minimize the system latency. In addition, game theory can effectively solve the decision-making problems of multiple rational players. Some studies use game theory to solve the optimization problems of computing offloading and resource allocation. In [26], the authors studied the problem of resource allocation in smart grid, and proposed a dynamic differential game model to maximum the benefits and resource utilization. In [27], the authors presented a UAV-assisted multi-access edge computing system, and proposed a game theory-based method to minimize the weighted cost of time delay and energy consumption. In [28], the authors studied the computation offloading problem in multi-carrier enabled mobile edge computing system, and proposed the cooperative game theory to minimize the total computation overhead. However,

these studies do not consider service caching in the edge computing-based smart grid.

Service caching effectively reduces the latency and energy consumption on acquiring services when related computing tasks are offloaded to ESs. Therefore, service caching is as important as computing offloading, and should attract more attention [29]–[31]. The authors in [32] considered the service caching in the edge computing-based smart grid, and proposed an online service caching algorithm to minimize the processing delay. In [33], the authors studied the problem of delay control strategy combining service caching and task offloading to effectively improve the quality of service. The authors in [34] investigated the collaborative caching and workload scheduling in edge computing, and proposed an iterative algorithm to reduce service response time and outsourcing traffic. In [35], the authors constructed the problem of offloading dependent tasks with considering service caching, and a convex programming based algorithm is proposed to minimize application's completion time. The authors in [36] optimized a joint caching, computing and communications problem which involves software multicasting, and converted it into an equivalent convex MINLP problem to minimize the weighted sum energy consumption. In [37], the authors considered the joint optimization of computing, service caching, and resource allocation, and proposed an efficiently approximate method based on semidefinite relaxation to minimize the system cost. The authors in [38] proposed a joint service caching placement and computation offloading decision optimization problem and transformed the original problem into a standard multidimensional knapsack problem to minimize the overall computing delay and energy consumption.

Although some studies have introduced edge computing into smart grid, the heterogeneity of computing tasks and service caching is often ignored. Generally speaking, the processing of computing tasks generated by SDs requires other services. Different from the above existing studies, this paper focuses on minimizing the weighted sum of task processing delay and energy consumption, and takes the delay constraint into consideration in the edge computing-based smart grid. Meanwhile, except for obtaining the offloading strategy of each SD, service caching is also an essential factor for addressing the minimization problem in edge computing-based smart grid. Therefore, we jointly consider the optimization of computing offloading and service caching in the edge computing-based smart grid. Furthermore, we decompose the optimization problem into an equivalent master problem and sub-problem, and design two low-complexity algorithms to address the corresponding problem, respectively. A gradient descent allocation algorithm is proposed to determine the resource allocation strategy, and then a game theory-based algorithm is proposed to obtain the computing strategy.

3 System Model

This section first presents the system architecture, and then gives the detailed definition of the communication model in the edge computing-based smart grid. Finally, we introduce the computing model.

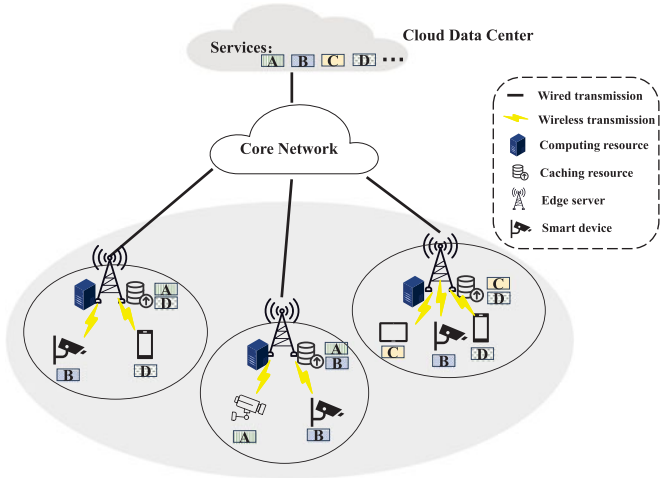


Fig. 1. System model.

3.1 System Architecture

As shown in Fig. 1, our system contains M ESs, I SDs, and a remote cloud server. Let $\mathcal{M} = \{1, 2..., M\}$ and $\mathcal{I} = \{1, 2..., I\}$ denote the set of ESs and SDs, respectively. Each SD can connect to its associated ES through a wireless link, and the ESs and cloud server are connected through optical fiber wired links. Each ES $m \in \mathcal{M}$ has limited computing resources F_m and cache capacity D_m, and it can provide computing offloading for SDs within its radio coverage. Cloud server can be regarded as a data center. Let $\mathcal{J} = \{1, 2..., J\}$ denote the set of computing task types. We assume that each SD generates only one computationally intensive task, which can be calculated locally or offloaded for remote processing, and each computing task is inseparable. For the computing task of type j in SD i, it can be described by triples as $H_j^i = \{C_j^i, D_j^i, T_{j,max}^i\}, i \in \mathcal{I}, j \in \mathcal{J}$. For task H_j^i, C_j^i represents the number of CPU cycles required to complete the task, D_j^i represents the data required by the computing task, $T_{j,max}^i$ represents the maximum delay that the task can tolerate.

Due to different data types, the computing input data D_j^i is composed of two components. Specifically, the data collected by the SD i is represented by S_j^i, and the relevant service of the computing task is represented by U_j. We use $Y_j^m \in \{0, 1\}$ to represent the caching strategy of ESs. When $Y_j^m = 0$, it means that ES m does not cache the relevant service, otherwise $Y_j^m = 1$. Each ES can cache services of different types of tasks and push its cached service to SDs. Due to the limited cache capacity, each ES cannot cache services of all types of tasks. If the ES does not cache the required service when receiving the offloaded task, it will initiate a request to the cloud server and download it. Our system architecture can be reflected or applied to many practical scenarios in the smart grid, such as surveillance and security monitoring systems, intelligent schesuling and fault monitoring and repair. In these scenarios, a large amount of sensors data and cameras data need to be transmitted to ES for analysis and computing steps. If the computing tasks exceed the load of the ESs, SDs can perform local computing or offload the task to the remote cloud for processing. This paper focuses more on edge computing, so we do not introduce the process of remote cloud processing in detail.

TABLE 1
Notations and Explanation

Notation	Explanation
$\mathcal{M}$	The set of ESs
$\mathcal{I}$	The set of SDs
F_m	The computing resources of the ES m
D_m	The caching capacity of the ES m
$\mathcal{J}$	The set of computing task types
C_j^i	The number of CPU cycles required to complete the task H_j^i
D_j^i	The size of data required to complete the task H_j^i
$T_{j,max}^i$	The time constraint of the task H_j^i
Y_m^j	The caching strategy of ES m
S_j^i	The input data of the task H_j^i collected by SD i
U_j^i	The relevant service of the task H_j^i
$R_{i,m}^U, R_{i,m}^D$	The uplink and downlink transmission rate between SD i and ES m
$X_{j,z}^i$	The offloading strategy of SD i
ζ_m^i	The computing strategy of SD i
r^{mc}	The transmission rate between the cloud server and the ES m
f_{loc}^i	The computing resource of SD i
f_m^i	The computing resource of ES m allocated to SD i
$T_{j,loc}^i, T_{j,m}^i$	The computing time for SD i in local/ES m processing
$E_{j,loc}^i, E_{j,m}^i$	The energy consumption for SD i in local/ES m processing
ζ	The set of computing strategy
$\mathcal{N}_{i,m}^{es}$	The set of SDs offload the task to ES m
$\mathcal{N}_{i,m}^{ld}$	The set of SDs process the task locally
$\mathcal{F}$	The computing resource allocation strategy for each ES

For ease of reference, the corresponding explanation of main notations is summarized in Table 1.

3.2 Communication Model

When a SD wants to upload the data collected by itself to the ES, it has to first consider the uplink data rate. In addition, we assume that a SD can only transmit data to one ES in a time slot. Let $\mathcal{N}_m$ denote the set of SDs in the ES m communication range. We denote $g_{i,m}$ and $d_{i,m}$ as the effective channel power gain coefficient and the distance from SD i to ES m, respectively. In this paper, we consider the situation where SDs are stationary during the data offloading process, so $g_{i,m}$ can be regarded as a constant in a time slot, then the corresponding channel power gain can be obtained as:

$$G_{i,m} = g_{i,m} d_{i,m}^{-\varepsilon}, \tag{1}$$

where ε is the path loss factor. Then, the uplink transmission rate of SD i which chooses to offload its collected data to the ES m through the wireless link can be expressed as:

$$R_{i,m}^U = Blog_2\left(1 + \frac{G_{i,m} p_i}{\Sigma_{k \in \mathcal{I} \setminus \{i\}} G_{k,m} p_k + \sigma^2}\right), \tag{2}$$

where B, p_i and σ^2 are the available spectral bandwidth, the uplink transmission power of SD i, and the noise power respectively. In the wireless interference model, ESs share the same spectrum resource.

Similarly, since the ES can send the cached relevant service (for example, library and database) required by the computing task to the corresponding SDs, we use p_m to denote the downlink transmission power of ES m, then the achievable transmission rate of SD i on the downlink is:

$$R_{i,m}^D = Blog_2\left(1 + \frac{G_{i,m} p_m}{\Sigma_{n \in \mathcal{M} \setminus \{m\}} G_{i,n} p_n + \sigma^2}\right), \tag{3}$$

It should be noted that the delay in transmitting the computing results is ignored in this paper because the size of the computing results is much smaller than the size of the input data.

3.3 Computing Model

According to the above system model, SDs can perform local computing or offload tasks to ESs. We use $X_{j,m}^i \in \{0,1\}, \forall m \in \mathcal{M}$ to represent the task offloading strategy. When $X_{j,m}^i = 0$, it means that the task is computed locally, otherwise it means that the task is offloaded to the ESs. We use $\zeta_m^i \in \{0,1,2,3\}$ to represent the computing strategy of SD i. In the following, we will describe these four situations in detail.

1) Local Computing Model. When $X_{j,m}^i = 0$, SD i completes the computing task H_j^i locally by its own CPU. It is assumed that the computing power of each SD remains unchanged during the calculation process. The computing capacity of SD i is expressed as f_{loc}^i. At this time, depending on whether the ES caches the relevant service of the task, the model of local computing can be divided into two situations:

When $\zeta_m^i = 0$, it indicates that SD i chooses to perform the computing locally, and ES m does not cache the relevant service. At this time, ES m needs to request a download from the cloud server and deliver the service to SD i, which completes the computing task locally. The relatively insignificant time the ES takes to initiate a request to the cloud server is ignored here. Then, the processing time of the task H_j^i by computing locally can be expressed by:

$$T_{j,loc}^i = \frac{C_j^i}{f_{loc}^i} + \frac{U_j}{R_{i,m}^D} + \frac{U_j}{r^{mc}}, \tag{4}$$

where $\frac{C_j^i}{f_{loc}^i}$ and $\frac{U_j}{R_{i,m}^D}$ indicate the computing delay of task completed locally by SD i, and the downlink transmission time between SD i and ES m, respectively. In addition, $\frac{U_j}{r^{mc}}$ is the transmission delay of the relevant service sent by the cloud server to ES m, and r^{mc} is the transmission rate between the cloud server and the ES. Since the cloud server and ESs are connected by the wired link, we set r^{mc} as a constant.

We denote e_{loc}^i as the energy consumption rate of local computing and p_c as the transmission energy consumption of each bit between ES m and cloud server. Then, the local processing energy consumption of the task H_j^i can be calculated as:

$$E_{j,loc}^i = e_{loc}^i \frac{C_j^i}{f_{loc}^i} + p_m \frac{U_j}{R_{i,m}^D} + p_c U_j. \tag{5}$$

When $\zeta_m^i = 1$, it indicates that SD i chooses to perform the computing locally, and ES m has cached the relevant service of the task. In this case, we can calculate the total computing delay of SD i to process the task H_j^i locally as:

$$T_{j,loc}^i = \frac{C_j^i}{f_{loc}^i} + \frac{U_j}{R_{i,m}^D}. \tag{6}$$

Since ES m has already cached the relevant service, it is no longer necessary to request it from the cloud sever. Then, the local processing energy consumption of the task H_j^i can be expressed by:

$$E_{j,loc}^i = e_{loc}^i \frac{C_j^i}{f_{loc}^i} + p_m \frac{U_j}{R_{i,m}^D}. \tag{7}$$

2) Edge Computing Model. When $X_{j,m}^i = 1$, it means that SD i offloads the task to ES m for computing. Let f_m^i represent the computing resources allocated by ES m to SD i. Similar to the model of the local computing, the model of edge computing can also be divided into two situations:

When $\zeta_m^i = 2$, it indicates that SD i decides to offload the computing task to the connected ES m, and ES m does not cache the relevant service required for the task, which needs to be requested and downloaded from the cloud server. In this case, the execution time of ES m for calculating the task H_j^i can be given as:

$$T_{j,m}^i = \frac{C_j^i}{f_m^i} + \frac{S_j^i}{R_{i,m}^U} + \frac{U_j}{r^{mc}}, \tag{8}$$

where $\frac{C_j^i}{f_m^i}$ is the computing delay for executing the computing task in ES m, $\frac{S_j^i}{R_{i,m}^U}$ is the uplink transmission time between SD i and ES m to transmit the data collected by the SD i.

We denote e_m^i as the power consumption rate of computation in the ES m. Then, the total energy consumption of the computing task offloaded to ES m can be obtained in the following way:

$$E_{j,m}^i = e_m^i \frac{C_j^i}{f_m^i} + p_i \frac{S_j^i}{R_{i,m}^U} + p_c U_j. \tag{9}$$

When $\zeta_m^i = 3$, it indicates that SD i decides to offload the computing task to the connected ES m, and ES m has cached the relevant service required for the task. In this case, the total computing delay of the task H_j^i by offloading to the connected ES m can be expressed by:

$$T_{j,m}^i = \frac{C_j^i}{f_m^i} + \frac{S_j^i}{R_{i,m}^U}. \tag{10}$$

Similarly, as ES m has cached the relevant service, there is no transmission energy consumption between ES m and the cloud server. Then the total corresponding energy consumption can be obtained by:

$$E_{j,m}^i = e_m^i \frac{C_j^i}{f_m^i} + p_i \frac{S_j^i}{R_{i,m}^U}. \tag{11}$$

4 Problem Formulation

In this paper, we define the task cost as the weighted sum of the task execution time and energy consumption. We use T_j^i and E_j^i to represent the actual completion time and energy consumption of the task, respectively. Therefore, the task cost of SD i can be written:

$$\varphi^i = \alpha T_j^i + (1-\alpha) E_j^i, \tag{12}$$

where α is the weight of the actual completion time. A larger α indicates that the computing task is more sensitive to the processing delay. On the contrary, the computing task is more sensitive to the energy consumption.

As mentioned above, when executing a computing task, each SD has four optional computing strategies, and it can choose one of them to minimize the task cost. Generally speaking, each SD tends to choose the computing strategy that can achieve the lowest task cost. Let ζ be the set of SDs' computing strategies, and $\mathcal{F}$ be the set of the resource allocation strategies. Therefore, the objective of this paper is to minimize the total task cost of the system by optimizing both the computing strategy and the computing resource allocation strategy, under the maximum delay constraints of tasks and the maximum cache capacity constraints of ESs, which is formulated as:

$$\min_{\zeta,\mathcal{F}} \varphi(\zeta, \mathcal{F}) = \sum_{i \in \mathcal{I}} \varphi^i \tag{13}$$

$$s.t. \sum_{i \in \mathcal{I}} f_m^i \le F_m, \forall m \in \mathcal{M}, \tag{14}$$

$$0 \le f_m^i \le F_m, \forall i \in \mathcal{I}, m \in \mathcal{M}, \tag{15}$$

$$f_{loc}^i \ge 0, \forall i \in \mathcal{I}, \tag{16}$$

$$T_j^i \le T_{j,max}^i, \forall i \in \mathcal{I}, \tag{17}$$

$$\sum_{j \in \mathcal{J}} U_j Y_j^m \le D_m, \forall m \in \mathcal{M}, \tag{18}$$

$$X_{j,m}^i \in \{0,1\}, \forall i \in \mathcal{I}, m \in \mathcal{M}, \tag{19}$$

$$Y_j^m \in \{0,1\}, \forall m \in \mathcal{M}, j \in \mathcal{J}, \tag{20}$$

$$\zeta_m^i \in \{0,1,2,3\}, \forall i \in \mathcal{I}, m \in \mathcal{M}. \tag{21}$$

Constraints (14) and (15) imply that each ES must allocate a positive computing resource to the associated SDs, and the total computing resources allocated to all the associated SDs should be less than the ES's idle computation resources. Constraint (16) indicates that the available computing resource should be non-negative. Constraint (17) indicates that the processing delay of the computing task should be less than the maximum tolerant delay. Constraint (18) describes that the sum of computing task relevant services cached at ES m should be less than or equal to the maximum cache capacity. Constraint (19) implies that each SD can offload to the ES or complete locally. Constraint (20) represents a binary caching strategy. Constraint (21) indicates that each SD i has four optional computing strategies.

In this form, the optimization problem has both binary variables and continuous non-integer variables. Therefore, the optimization problem is a Mixed Integer Nonlinear Programming (MINLP), which belongs to NP-hard. In addition, due to the coupling between offloading strategy variable and caching strategy variable, the objective function in problem (13) is discrete and non-convex. Therefore, it is difficult to find the optimal solution of this problem in polynomial time.

5 Problem Decomposition

In this section, we first modify the constraints of the original problem appropriately so that the constraints on the computing strategy ζ and the computing resource allocation

strategy $\mathcal{F}$ can be separated from each other. Then, we decompose the original optimization problem into an equivalent master problem and sub-problem to minimize the total task cost of the system.

We first consider the scenario that ESs do not cache the relevant services, and generate an initial computing strategy that satisfies the above constraints. Under this situation, we can temporarily ignore constraint (18) to decouple problem (13). According to Eq. (8), Constraints (14), (15) and (17) can be rewritten as follows:

$$\sum_{i \in \mathcal{N}_{i,m}^{es}} f_m^{i,min} \leq F_m, \forall m \in \mathcal{M} \tag{22}$$

$$f_m^i \geq f_m^{i,min}, \forall i \in \mathcal{N}_{i,m}^{es}, m \in \mathcal{M} \tag{23}$$

where $\mathcal{N}_{i,m}^{es}$ is the set of SDs which need to offload the task to the associated ES m, and $f_m^{i,min} = C_j^i/(T_{j,max}^i - S_j^i/R_{i,m}^U - U_j/r^{mc})$ represents the minimum computing resources that need to be provided if SD i offloads its task to ES m, according to Eq. (8). Since $f_m^{i,min}$ can meet the maximum tolerable delay of SD i when the ES does not cache the relevant service, it is obvious that it can also meet the maximum tolerable delay of SD i when the ES caches the related service. The caching strategy may change in subsequent iterations. If $Y_j^m = 1$, $f_m^{i,min} = C_j^i/(T_{j,max}^i - S_j^i/R_{i,m}^U)$ according to Eq. (10). If $i \notin \mathcal{N}_{i,m}^{es}$, then $f_m^{i,min} = 0$.

Hence, if ζ, $\mathcal{F}$ satisfy Constraints (22) and (23), the Constraints (15) and (17) are also satisfied. Thus, we can rewrite the original problem as follows:

$$\min_{\zeta} \left(\min_{\mathcal{F}} \varphi(\zeta, \mathcal{F}) \right) \tag{24}$$

$$s.t.\ (16) - (22), \tag{25}$$

$$(14), (23). \tag{26}$$

It is worth noting that the constraints on the computing strategy ζ in (25), and the computing resource allocation strategy $\mathcal{F}$ in (26), are decoupled from each other. Therefore, solving the problem in (24) is equivalent to solve the following problem:

$$\min_{\zeta} \varphi^*(\zeta) \tag{27}$$

$$s.t.\ (16) - (22), \tag{28}$$

where $\varphi^*(\zeta)$ corresponds to the optimal-value function when the resource allocation strategy $\mathcal{F}$ takes the optimal solution under the fixed computing strategy $\tilde{\zeta}$, written as:

$$\varphi^*(\zeta) = \min_{\mathcal{F}} \varphi(\tilde{\zeta}, \mathcal{F}) \tag{29}$$

$$s.t.\ (14), (23). \tag{30}$$

In the next section, we will propose two algorithms to obtain the sub-optimal computing resource allocation strategy $\mathcal{F}$ and computing strategy ζ.

6 Main Approach

This section presents a Collaborative Computing Offloading and Resource Allocation Method (CCORAM), which includes two low-complexity algorithms to solve the above optimization problem. A gradient descent allocation algorithm is proposed to determine the resource allocation strategy, then a game theory-based algorithm is proposed to obtain the computing strategies of SDs.

6.1 Computing Resource Allocation

This part proposes a gradient descent allocation algorithm to allocate computing resources to each SD that offloads tasks to the ESs. The goal is to minimize the cost of computing tasks by allocating computing resources appropriately.

Once a feasible computing strategy $\tilde{\zeta}$ that satisfies the constraints (25) is given, the objective function in (29) can be expressed as:

$$\varphi(\tilde{\zeta}, \mathcal{F}) = \varphi_{loc}(\tilde{\zeta}) + \varphi_{es}(\tilde{\zeta}, \mathcal{F}), \tag{31}$$

$$= \sum_{i \in \mathcal{N}_{i,m}^{loc}} \varphi^i + \sum_{i \in \mathcal{N}_{i,m}^{es}} \varphi^i.$$

where $\mathcal{N}_{i,m}^{loc}$ is the set of SDs which choose to perform the computing locally. In Eq. (31), the first item represents the task cost of SDs which select local computing in the current computing strategy, and the second item represents the task cost of SDs which offload their tasks to the associated ESs. Obviously, the first item of Eq. (31) is constant when the computing strategy $\tilde{\zeta}$ is fixed. Therefore, problem (29) can be transferred as:

$$\min_{\mathcal{F}} \varphi_{es}(\tilde{\zeta}, \mathcal{F}) \tag{32}$$

$$s.t.\ (14), (23). \tag{33}$$

Next, the second derivative of φ can be calculated according to Eq. (31), which is expressed as:

$$\frac{\partial^2 \varphi_{es}}{\partial^2 f_m^i} > 0, \forall i \in \mathcal{N}_{i,m}^{es}, m \in \mathcal{M} \tag{34}$$

$$\frac{\partial^2 \varphi_{es}}{\partial f_m^i \partial f_n^k} = 0, \forall (i,m) \neq (k,n). \tag{35}$$

where the Hessian matrix of $\varphi_{es}(\tilde{\zeta}, \mathcal{F})$ is diagonal and has strictly positive elements, so it is positive-definite. Therefore, problem (32) is convex and can be solved by Lagrangian duality and Karush-Kuhn-Tucker (KKT) conditions. However, solving the multivariable system of nonlinear equations will bring high complexity.

In order to reduce the complexity of the algorithm, we design Algorithm 1 to solve this sub-problem. First, we divide the computing resources of each ES into a number of small atomic chunks, denoted by β. Then, we allocate these computing resource chunks one by one to SDs which offload computing tasks to the ES. The key point of the algorithm is to allocate computing resources to the SD with the fastest change (the gradient is negative and the minimal). It is worth noting that the smaller the division of computing resources, the closer the final solution is to the optimal solution. More detailed information is shown in Algorithm 1.

6.2 Game Theory-Based Computing Strategy

In the computing strategy-making problem, the computing strategy of SD i not only depends on its own offloading requirements but also on the computing strategies of other SDs. Game theory is a mathematical theory and method for

studying phenomena of a struggle or competitive nature. The inherent competitiveness of the computing strategy problem shows that game theory is feasible to solve the above problem. Therefore, a game theory-based algorithm is used to solve the problem of computing strategy-making.

In this part, the problem of computing strategy-making is modeled as a non-cooperative game. In the game, players are all SDs, which are competing for resources to minimize the total task cost. The game can be defined as $\Gamma = (\mathcal{I}, \{\zeta^i\}_{i\in\mathcal{I}}, \{\varphi^i\}_{i\in\mathcal{I}})$, where $\mathcal{I}$ is the set of the rational game players, $\{\zeta^i\}$ is the set of strategies for each player, and $\{\varphi^i\}$ represents SD i's task cost function. Each SD tries to select a profitable computing strategy to minimize the task cost. Then, the important concept of Nash Equilibrium (NE) is introduced to solve the proposed computing strategy game [39].

Algorithm 1. Gradient Descent Allocation Algorithm

Require: feasible computing strategy $\tilde{\zeta}$ and unit computing resource size β;
Ensure: $\mathcal{F}$;
for each ES $m \in \mathcal{M}$ **do**
  for each SD $i \in \mathcal{N}_{i,m}^{es}$ **do**
    $f_m^i \leftarrow f_m^{i,min}$
  end for
  $f_m^{left} \leftarrow F_m - \sum_{i\in\mathcal{N}_{i,m}^{es}} f_m^i$
  $v_m \leftarrow \lfloor f_m^{left}/\beta \rfloor$
end for
for each ES $m \in \mathcal{M}$ **do**
  $\sigma \leftarrow \frac{\partial \varphi_{es}}{\partial f_m^i}$
  $G = \{g_i = \sigma(f_m^i), \forall i \in \mathcal{N}_{i,m}^{es}\}$
  $count \leftarrow 0$
  while $count < v_m$ **do**
    $i \leftarrow argmin_{i\in\mathcal{N}_{i,m}^{es}}\{G\}$
    $f_m^i \leftarrow f_m^i + \beta$
    $g_i \leftarrow \sigma(f_m^i)$
    $count \leftarrow count + 1$
  end while
end for
return $\mathcal{F}$

NE of strategy-making problem is a sub-optimal computing strategy profile $\zeta_m^* = (\zeta_m^{1*}, \zeta_m^{2*}, \ldots, \zeta_m^{i*}, \ldots, \zeta_m^{I*})$, where ζ_m^{i*} is the sub-optimal computing strategy formulated when SD i knows the computing strategies of other SDs. Let $\zeta_m^{-i} = (\zeta_m^1, \zeta_m^2, \zeta_m^3, \ldots \zeta_m^{i-1}, \zeta_m^{i+1}, \ldots)$ be the computing strategies for all other SDs except SD i, then we have:

Definition 1. *A strategy ζ_m^* is the NE for the game, if for any $i \in \mathcal{I}$, we have the following relationship:*

$$\varphi(\zeta_m^{i*}, \zeta_m^{-i}) \leq \varphi(\zeta_m^i, \zeta_m^{-i}), \forall i \in \mathcal{I}. \tag{36}$$

NE is the stable state of the system. In this state, no SD can unilaterally leave the stable state to reduce more task costs. According to the NE existence theorem, for games with a restricted number of participants (each participant has restricted optional strategies), NE can converge in a finite number of strategies [40].

After the resource allocation step, the computing strategies are updated by the game until obtaining NE. In addition, computing resource allocation needs to be optimized to minimize the task cost of all offloading SDs after each SD submits a computing strategy. By iterating over each other, the system enters a steady state.

Then, the details of the proposed game theory-based computing strategy are shown in Algorithm 2. First, an initial computing strategy ζ_Θ which satisfies the constraints is selected. Here, in order to decouple the original optimization problem, we assume that ESs do not cache the relevant services in the initial stage. Second, each SD will update the computing strategy based on the current strategy to reduce the task cost. Each SD randomly selects a computing strategy $\zeta_m^{i'}$ from Ψ, which is different from its initial computing strategy, and calculates the uplink transmission rate and downlink transmission rate according to Eq. (2) and Eq. (3). Then, the optimal computing resource allocation strategy $\mathcal{F}$ is obtained based on Algorithm 1. By comparing the value of $\varphi(\zeta_m^{i'}, \zeta_m^{-i})$ and $\varphi(\zeta_m^i, \zeta_m^{-i})$, each SD updates its marginal task cost $\varphi_{marginal}^i$ accordingly. If $\varphi(\zeta_m^{i'}, \zeta_m^{-i})$ is less than $\varphi(\zeta_m^i, \zeta_m^{-i})$, and the selected computing strategy satisfies the constraints (17) and (18), the marginal task cost of SD i is calculated as $\varphi(\zeta_m^i, \zeta_m^{-i}) - \varphi(\zeta_m^{i'}, \zeta_m^{-i})$. Finally, each SD competes to update its computing strategy opportunistically. In order to meet the game requirements, a SD with the maximum marginal task cost $\varphi_{marginal}^i$ will be selected to update its computing strategy. In each iterative process, for all SDs, the total task cost is computed separately based on the current computing strategy. The above process is repeated to reach the NE for a limited number of times, and a sub-optimal computing strategy configuration $\zeta = \{\zeta_1^*, \zeta_2^*, \ldots, \zeta_M^*\}$ can be obtained finally.

Algorithm 2. Game Theory-Based Computing Strategy Algorithm

Require: SD set $\mathcal{I} = \{1, 2, \ldots I\}$, computing task $H_j^i = \{C_j^i, D_j^i, T_{j,max}^i\}, i \in \mathcal{I}, j \in \mathcal{J}$, S_j^i, U_j, r^{mc}, e_{loc}^i, e_m^i;
Ensure: sub-optimal computing strategy profile;
1: **for** each ES $m \in \mathcal{M}$ **do**
2: $Y_j^m = 0, \forall m \in \mathcal{M}, j \in \mathcal{J}$;
3: Initial computing strategy ζ_Θ;
4: $\zeta_m^* \leftarrow \zeta_\Theta$;
5: **for** each SD $i \in \mathcal{N}_m$ **do**
6: $\Psi \leftarrow \{0, 1, 2, 3\} \backslash \zeta_m^i$;
7: Randomly select a computing strategy $\zeta_m^{i'}$ from Ψ;
8: Calculate $R_{i,m}^U$, $R_{i,m}^D$ according to Eqs. (2) and (3), and calculate $\mathcal{F}$ based on Algorithm 1;
9: **if** $\varphi(\zeta_m^{i'}, \zeta_m^{-i}) < \varphi(\zeta_m^i, \zeta_m^{-i})$ **and** satisfy the constraints: $T_j^i \leq T_{j,max}^i$, $\sum_{j=1}^J U_j Y_j^m \leq D_m$ **then**
10: calculate $\varphi_{marginal}^i = \varphi(\zeta_m^i, \zeta_m^{-i}) - \varphi(\zeta_m^{i'}, \zeta_m^{-i})$;
11: **end if**
12: **end for**
13: **for** each SD $i \in \mathcal{N}_m$ **do**
14: **if** $\varphi_{marginal}^i$ is the maximum **then**
15: update $\zeta_m^i = \zeta_m^{i'}$ into ζ_m^*;
16: **end if**
17: **end for**
18: **while** $\zeta_m^* \neq \zeta_\Theta$ **do**
19: $\zeta_\Theta \leftarrow \zeta_m^*$;
20: Repeat Lines 5−17;
21: **end while**
22: **end for**

6.3 Complexity Analysis

In Algorithm 1, we denote f_m^{left} as the remaining computing resources of ES m under a fixed computing strategy $\tilde{\zeta}$, and $\mathcal{M}$ as the set of ESs. The complexity of Algorithm 1 depends on $|\mathcal{M}|$ and the number of computing resources chunks v_m (which is calculated by f_m^{left} and β). The loop complexity of the inner layer is $O(v_m)$, while the complexity of the outer layer is $O(|\mathcal{M}|)$. In summary, we can know that the complexity of Algorithm 1 is $O(|\mathcal{M}| \times v_m)$. The while loop in Algorithm 2 needs K iterations to converge, thus the complexity of the Algorithm 2 is $O(K|\mathcal{N}_m|(|\mathcal{M}| \times v_m))$. For the Enumeration Method (EM), however, all the computing strategy combinations need to be traversed and the running time is $O(4^{|\mathcal{N}_m|})$. Therefore, the complexity of the EM is $O(4^{|\mathcal{N}_m|}(|\mathcal{M}| \times v_m))$.

7 Simulation Results

In this section, we conduct simulations to evaluate the performance of the proposed method.

7.1 Parameters Settings

We consider a scenario consisting of 3 ESs, 10 SDs and a remote cloud. The radius of each ES is set to 200 m. In addition, each SD is randomly distributed in the communication range of 3 ESs, and communicates with the associated ES through a wireless channel. The wireless channel gain $G_{i,m}$ is modeled as $g_{i,m}d_{i,m}^{-\varepsilon}$, where $g_{i,m}$ is set as a constant, $d_{i,m}^{-\varepsilon}$ represents the distance between the SD i and its associated ES m, and the path loss coefficient is $\varepsilon = 4$. The uplink transmission power of each SD is set to 0.01 W. In addition, the downlink transmission power of each ES is 2 W. The total channel bandwidth is set to 20 MHz, and the noise power $\sigma^2 = 2 \times 10^{-13}$ W [41].

We assume that there are 6 different types of computing tasks. It should be noted that processing different types of computing tasks requires different sizes of input data and CPU cycles. For example, the collected data of smart monitoring application by SDs is set between 2 MB and 8 MB, and the number of CPU cycles is set between 1000 Megacycles and 4000 Megacycles [42]. The caching capacity required by the service is uniformly distributed from 30 MB to 80 MB. The computing capacity f_{loc}^i is set equally for each SD, e.g., $f_{loc}^i = 1024$ Megacycles/sec. For each ES, we assume that the idle computing resources is in the range of [10, 30] GHz, and the caching capacity is 100 MB.

In order to evaluate the proposed method, we compare it with the following four benchmark methods [43]:

- *Enumeration Method (EM):* The method traverses all computing strategy combinations, and allocates computing resources according to our proposed Algorithm 1.
- *Game theory-based Computing Strategy with Popularity Caching (GCSPC):* Each ES caches relevant services based on the popularity of contents, and SDs then select computing strategies based on game theory.
- *Game theory-based Computing Strategy with Random Caching (GCSRC):* Each ES caches relevant services randomly until the cache capacity is full, and SDs then select computing strategies based on game theory.
- *Game theory-based Computing Strategy without Caching (GCSC):* Each ES does not cache relevant services, and SDs select computing strategies based on game theory.

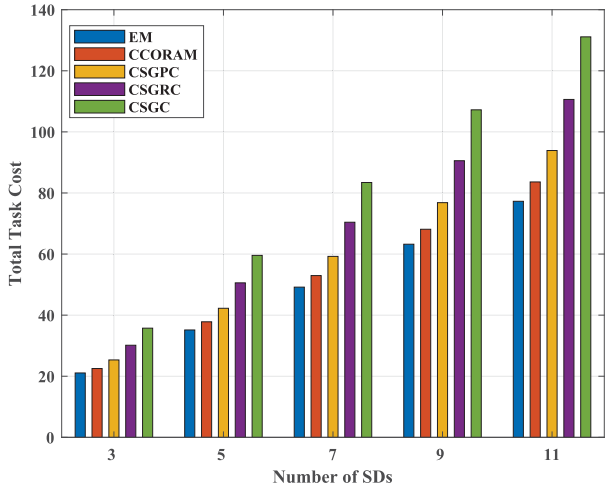


Fig. 2. Performance comparison in terms of the total task cost with different number of SDs.

7.2 Performance Comparison

In this part, we compare the proposed CCORAM with other benchmark methods in terms of the total task cost under different scenarios.

Fig. 2 shows the performance comparison of EM, CCORAM, GCSPC, GCSRC, and GCSC in terms of the total task cost when the network is sparse (the number of SDs increases from 3 to 11). It can be found that with the increases of the number of SDs, more computing tasks will be generated and processed, so the total task cost will increase continuously. EM performs as an upper bound to all other methods certainly, which is approximately the optimal solution. However, the time complexity of EM is very high if the number of SDs is large, since it needs to traverse all computing strategy combinations. As shown in Fig. 2, the total task cost of our proposed method is very close to that of EM, but the time complexity of CCORAM is much lower than that of EM as analyzed above. Therefore, CCORAM is a sub-optimal method, which can achieve near-optimal solution with low time complexity. Among all the other methods, CCORAM always has the best performance in terms of the total task cost with the increasing number of SDs. This is because CCORAM enables ESs to cache more suitable relevant services, and makes SDs select more suitable computing strategies. As for GCSC, its task cost is higher than other four methods because it does not consider service caching.

Fig. 3 shows the performance comparison of EM, CCORAM, GCSPC, GCSRC, and GCSC in terms of the total task cost under different caching capacities of ESs. Although EM can obtain the optimal solution, its enormous time complexity makes the simulation only available in the sparse network. Therefore, we set the number of ESs as $M = 3$, and the number of SDs as $I = 10$. It can be found the

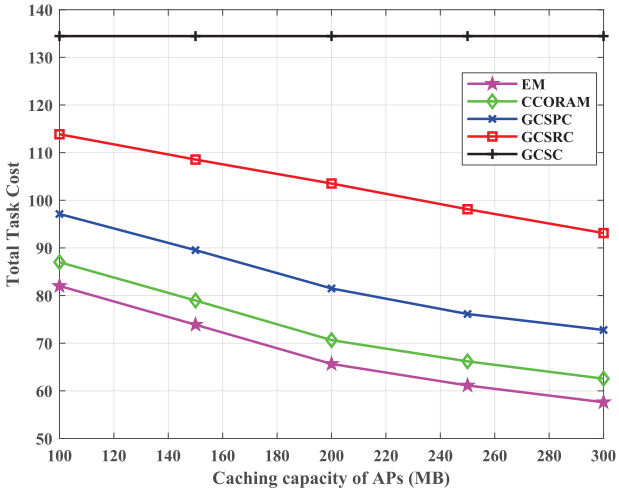


Fig. 3. Performance comparison in terms of the total task cost with different caching capacity of ESs.

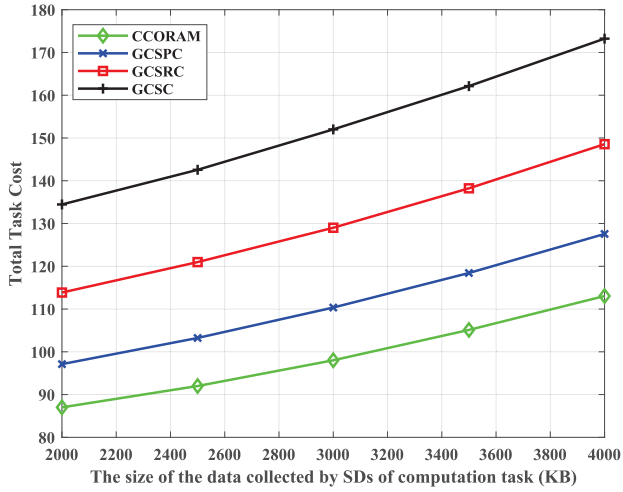


Fig. 5. Performance comparison in terms of the total task cost with different sizes of data collected by SDs.

performance of CCORAM is very close to the optimal solution EM, and its performance is significantly better than other benchmark methods. The total task cost of GCSC remains unchanged with the increase of ES's caching capacity, and is higher than other methods. This is because if the ES has not cached the relevant service when receiving an offloaded task, it will request and download from the cloud server, thus increasing the task cost. It is worth noticing when the caching capacity of ESs is increased, the required cost for task processing in EM, CCORAM, GCSPC and GCSRC are reduced. This is because when the caching capacity of ESs is increased, they can cache more relevant services. Then, more tasks do not need to request the relevant service from the cloud server, thus reducing the total task cost from both latency and energy consumption.

Fig. 4 shows the performance comparison of CCORAM, GCSPC, GCSRC and GCSC in terms of the total task cost when the network is dense. We set the number of ESs as $M = 3$, and the number of SDs as $I = 30$. Due to the enormous time complexity, EM is omitted in this part. The total task cost of CCORAM, GCSPC, GCSRC and GCSC all decrease with the increase of the ESs' computing capacities. This is because more computing resources in ESs contribute to lower delay, thus resulting in better rewards. That is, in the CCORAM, GCSPC, GCSRC and GCSC, the delay induced for task processing is reduced when increasing the computing resources of ESs. Moreover, we can see that the total task cost of CCORAM is always smaller than the other three benchmark methods. This is because CCORAM can obtain a better caching strategy and reduce the total task cost under the same computing resource allocation method.

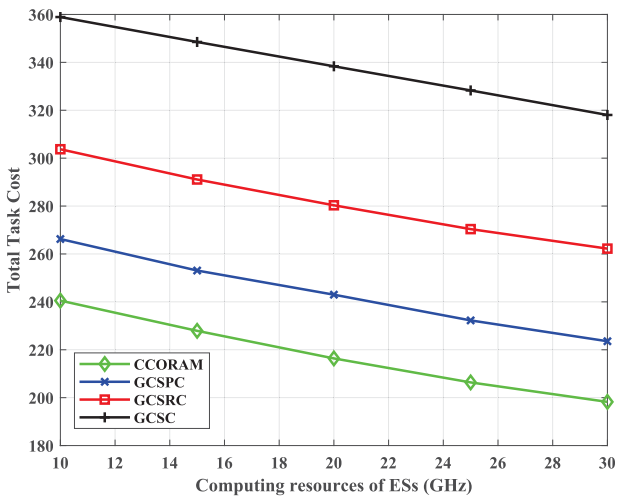


Fig. 4. Performance comparison in terms of the total task cost with different computing resources of ESs.

Fig. 5 shows the performance comparison of CCORAM, GCSPC, GCSRC, and GCSC in terms of the total task cost under different sizes of data collected by SDs. We set the number of ESs as $M = 3$, and the number of SDs as $I = 10$. It can be found that the total task cost of CCORAM, GCSPC, GCSRC, and GCSC all increase with the increase of the data collected by the SDs. This is because with the increase of data collected by SDs, the transmission time and energy consumption of SDs offloading tasks to ESs will also increase. Moreover, when the data collected by SDs is too large, most SDs can switch to local execution instead of offloading the task to ES. Under this situation, the proposed CCORAM still performs best, and reduces the total task cost by 11.38%, 24%, and 34.74% compared with GCSPC, GCSRC and GCSC, respectively. This is because our proposed CCORAM takes full advantage of ESs' resources and has more flexibility to design SDs' computing strategy.

To summarize, the computing resources F_m and caching capacity D_m have significant impact on the performance of CCORAM, and CCORAM is very close to the optimal EM and significantly outperforms the other benchmark methods under different scenarios. Moreover, it cannot be ignored that CCORAM is better than EM in terms of time complexity.

7.3 The Convergence Performance

In this part, we analyze the convergence of our proposed method.

Fig. 6 shows the convergence of the while loop of the CCORAM under the different computing resources of the

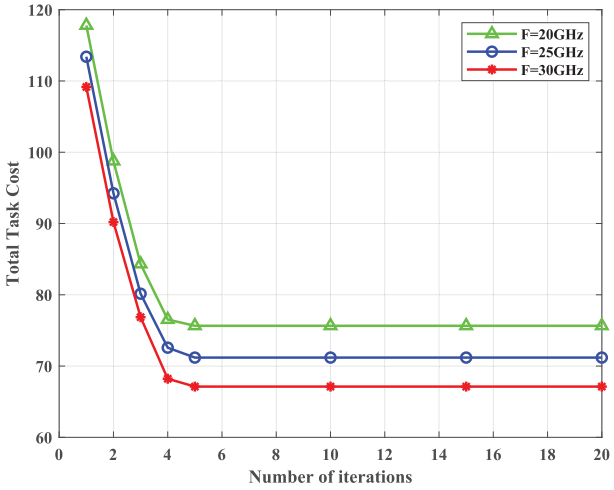


Fig. 6. Convergence of the CCORAM.

ESs. As shown in Fig. 6, the total task cost of our proposed method is relatively high at the beginning and decreases rapidly with the increase of the iterations, and stabilizes after about 10 while loop iterations, which demonstrates that our proposed method can converge quickly. The reason is that the CCORAM minimizes the total task cost in the iteration by optimizing computing strategy and computing resource allocation. With the increase of ESs' computing resources, more computing resources can be allocated to SDs which choose to offload their tasks, and the total task cost will decrease accordingly.

To summarize, it is proved that after a limited number of iterations, the system will reach NE, which further proves that the proposed CCORAM can not only approach the optimal total task cost with lower time complexity than EM, but also has stability.

8 Conclusion

This paper has investigated the joint optimization of computing offloading and service caching in the edge computing-based smart grid. First, we formalized the optimization problem as a Mixed-Integer Non-Linear Program (MINLP), aiming to minimize the task cost of the system. In order to solve the optimization problem, we decomposed the original optimization problem into an equivalent master problem and sub-problem, and proposed a Collaborative Computing Offloading and Resource Allocation Method (CCORAM), which includes two low-complexity algorithms. Simulation results show that CCORAM is very close to the optimal EM with lower time complexity, and significantly outperforms the other benchmark methods under different scenarios. In the future, we intend to consider collaboration between ESs to make full use of the caching and computing resources of idle ESs.

References

[1] J. Liu, J. Weng, A. Yang, Y. Chen, and X. Lin, "Enabling efficient and privacy-preserving aggregation communication and function query for fog computing-based smart grid," *IEEE Trans. Smart Grid*, vol. 11, no. 1, pp. 247–257, Jan. 2020.

[2] N. Cheng *et al.*, "Space/Aerial-assisted computing offloading for IoT applications: A. learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.

[3] C. Zhou *et al.*, "Deep reinforcement learning for delay-oriented IoT task scheduling in space-air-ground integrated network," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 911–925, Feb. 2021.

[4] K. Wang *et al.*, "A survey on energy internet: Architecture, approach, and emerging technologies," *IEEE Syst. J.*, vol. 12, no. 3, pp. 2403–2416, Sep. 2018.

[5] T. Qiu, K. Zheng, H. Song, M. Han, and B. Kantarci, "A local-optimization emergency scheduling scheme with self-recovery for a smart grid," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 3195–3205, Dec. 2017.

[6] H. Zhou, X. Chen, S. He, C. Zhu, and V. C. M. Leung, "Freshness-aware seed selection for offloading cellular traffic through opportunistic mobile networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 4, pp. 2658–2669, Apr. 2020.

[7] H. Zhou, T. Wu, X. Chen, S. He, and J. Wu, "RAIM: A reverse auction-based incentive mechanism for mobile data offloading through opportunistic mobile networks," *IEEE Trans. Netw. Sci. Eng.*, vol. PP, no. 99, p. 1, 2022, doi: 10.1109/TNSE.2021.3126367.

[8] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.

[9] H. Zhou, T. Wu, H. Zhang, and J. Wu, "Incentive-driven deep reinforcement learning for content caching and D2D offloading," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2445–2460, Aug. 2021.

[10] R. Deng, G. Xiao, R. Lu, and J. Chen, "Fast distributed demand response with spatially and temporally coupled constraints in smart grid," *IEEE Trans. Ind. Informat.*, vol. 11, no. 6, pp. 1597–1606, Dec. 2015.

[11] H. Zhou, X. Chen, S. He, J. Chen, and J. Wu, "DRAIM: A novel delay-constraint and reverse auction-based incentive mechanism for WiFi offloading," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 4, pp. 711–722, Apr. 2020.

[12] F. Zhang, R. Deng, X. Zhao, and M. M. Wang, "Load balancing for distributed intelligent edge computing: A state-based game approach," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 4, pp. 1066–1077, Dec. 2021.

[13] W. Chang and J. Wu, "Progressive or conservative: Rationally allocate cooperative work in mobile social networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 7, pp. 2020–2035, Jul. 2015.

[14] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, "Deep reinforcement learning for energy efficient computation offloading in mobile edge computing" *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1517–1530, Jan. 2022.

[15] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Joint service placement and request routing in multi-cell mobile edge computing networks," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 10–18.

[16] Y. Zhang, X. Lan, Y. Li, L. Cai, and J. Pan, "Efficient computation resource management in mobile edge-cloud computing," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3455–3466, Apr. 2019.

[17] S. Zhao *et al.*, "Smart and practical privacy-preserving data aggregation for fog-based smart grids," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 521–536, 2021.

[18] T. Sirojan, S. Lu, B. T. Phung, and E. Ambikairajah, "Embedded edge computing for real-time smart meter data analytics," in *Proc. Int. Conf. Smart Energy Syst. Technol.*, 2019, pp. 1–5.

[19] R. El-Awadi, A. Fernández-Vilas, and R. P. Díaz Redondo, "Fog computing solution for distributed anomaly detection in smart grids," in *Proc. IEEE Int. Conf. Wireless Mobile Comput. Netw. Commun.*, 2019, pp. 348–353.

[20] N. Kumar, S. Zeadally, and J. J. P. C. Rodrigues, "Vehicular delay-tolerant networks for smart grid data management using mobile edge computing," *IEEE Commun. Mag.*, vol. 54, no. 10, pp. 60–66, Oct. 2016.

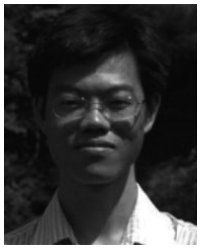
[21] Y. Huang, Y. Lu, F. Wang, X. Fan, J. Liu, and V. C. M. Leung, "An edge computing framework for real-time monitoring in smart grid," in *Proc. IEEE Int. Conf. Ind. Internet*, 2018, pp. 99–108.

[22] J. Jiang, J. Xu, Y. Xie, Y. Zhu, Z. Li, and C. Yang, "A cooperative computation offloading scheme for dense wireless sensor-assisted smart grid networks," in *Proc. IEEE Int. Conf. Comput. Commun. Secur.*, 2021, pp. 887–892.

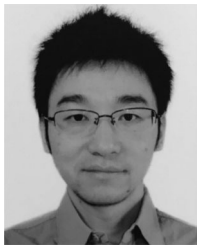
[23] O. Boudia, and S. Senouci, "An efficient and secure multidimensional data aggregation for fog-computing-based smart grid," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6143–6153, Apr. 2021.
[24] Q. Li, Y. Deng, W. Sun, and W. Li, "Communication and computation resource allocation and offloading for edge intelligence enabled fault detection system in smart grid," in *Proc. IEEE Int. Conf. Commun. Control Comput. Technol. Smart Grids*, 2020, pp. 1–7.
[25] S. Zahoor, N. Javaid, A. Khan, B. Ruqia, F. J. Muhammad, and M. Zahid, "A cloud-fog-based smart grid model for efficient resource utilization," in *Proc. IEEE 14th Int. Wireless Commun. Mobile Comput. Conf.*, 2018, pp. 1154–1160.
[26] Z. Li, Y. Liu, R. Xin, L. Gao, X. Ding, and Y. Hu, "A dynamic game model for resource allocation in fog computing for ubiquitous smart grid," in *Proc. IEEE 28th Wireless Opt. Commun. Conf.*, 2019, pp. 1–5.
[27] K. Zhang, X. Gui, D. Ren, and D. Li, "Energy-latency tradeoff for computation offloading in UAV-assisted multi-access edge computing system," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6709–6719, Apr. 2021.
[28] Q. Pham, H. T. Nguyen, Z. Han, and W. Hwang, "Coalitional games for computation offloading in NOMA-Enabled multi-access edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 1982–1993, Feb. 2020.
[29] K. Jiang, C. Sun, H. Zhou, X. Li, M. Dong, and V. C. M. Leung, "Intelligence-empowered mobile edge computing: Framework, issues, implementation, and outlook," *IEEE Netw.*, vol. 35, no. 5, pp. 74–82, Sep./Oct. 2021.
[30] Z. Ning *et al.*, "Intelligent edge computing in internet of vehicles: A. joint computation offloading and caching solution," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2212–2225, Apr. 2021.
[31] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE INFOCOM*, pp. 207–215, 2018.
[32] M. Li, L. Rui, X. Qiu, S. Guo, and X. Yu, "Design of a service caching and task offloading mechanism in smart grid edge network," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf.*, 2019, pp. 249–254.
[33] L. Li, and H. Zhang, "Delay optimization strategy for service cache and task offloading in three-tier architecture mobile edge computing system," *IEEE Access*, vol. 8, pp. 170211–170224, 2020.
[34] X. Ma, A. Zhou, S. Zhang, and S. Wang, "Cooperative service caching and workload scheduling in mobile edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 2076–2085.
[35] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang, "Offloading dependent tasks in mobile edge computing with service caching," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 1997–2006.
[36] W. Wen, Y. Cui, T. Q. S. Quek, F. Zheng, and S. Jin, "Joint optimal software caching, computation offloading and communications resource allocation for mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7879–7894, Jul. 2020.
[37] G. Zhang, S. Zhang, W. Zhang, Z. Shen, and L. Wang, "Joint service caching, computation offloading and resource allocation in mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 5288–5300, Aug. 2021.
[38] S. Bi, L. Huang, and Y. A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4947–4963, Jul. 2020.
[39] S. Tadelis, "Game theory: An introduction" *Econ. Books*, vol. 1. Princeton, NJ, USA: Princeton Univ. Press, 2012.
[40] H. Guo, J. Liu, and J. Zhang, "Efficient computation offloading for multi-access edge computing in 5G HetNets," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–6.
[41] X. Yang, Z. Fei, J. Zheng, N. Zhang, and A. Anpalagan, "Joint multi-user computation offloading and data caching for hybrid mobile cloud/edge computing," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11018–11030, Nov. 2019.
[42] Z. Lin, S. Bi, and Y. -J. A. Zhang, "Optimizing AI service placement and computation offloading in mobile edge intelligence systems," in *Proc. IEEE Glob. Commun. Conf.*, 2020, pp. 1–7.
[43] J. Tang, H. Tang, X. Zhang, K. Cumanan, G. Chen, and K. Wong, "Energy minimization in D2D-Assisted cache-enabled Internet of Things: A deep reinforcement learning approach," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5412–5423, Aug. 2020.

Huan Zhou (Member, IEEE) received the PhD degree from the Department of Control Science and Engineering, Zhejiang University. He was a visiting scholar with the Temple University from Nov. 2012 to May, 2013, and a CSC supported postdoctoral fellow with the University of British Columbia from November 2016 to November 2017. Currently, he is a full professor with the College of Computer and Information Technology, China Three Gorges University. He was a lead guest editor of Pervasive and Mobile Computing, TPC chair of EAI BDTA 2020, local arrangement chair of I-SPAN 2018, special session chair of the 3rd International Conference on Internet of Vehicles (IOV 2016), and TPC member of IEEE Globecom, ICC, ICCCN, etc. He has published more than 50 research papers in some international journals and conferences, including the *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Wireless Communications* and so on. His research interests include opportunistic mobile networks, VANETs, mobile data offloading, and mobile edge computing. He received the Best Paper Award of I-SPAN 2014 and I-SPAN 2018, and is currently serving as an associate editor of *IEEE Access* and *EURASIP Journal on Wireless Communications and Networking*.

Zhenyu Zhang (Student Member, IEEE) received the BS degree from Qingdao University. He is currently working toward the graduate degree in the College of Computer and Information Technology, China Three Gorges University. His main research interests include edge computing, computation offloading, and service caching.



Dawei Li (Member, IEEE) received the bachelor's degree from the Department of Electronics and Information Engineering (currently School of Electronic Information and Communications), Huazhong University of Science and Technology, Wuhan, China, in 2011, and the PhD degree from the Department of Computer and Information Sciences, Temple University, Philadelphia, PA, USA, in 2016. He is currently an assistant professor with the Department of Computer Science, Montclair State University, Montclair, NJ, USA. He has published research works in the IEEE INFOCOM, the *IEEE Transactions on Parallel and Distributed Systems*, and the *IEEE Transactions on Computers*. His research interests include the general fields of parallel and distributed systems, including green computing, data center networks, and cloud and edge computing.



Zhou Su (Senior Member, IEEE) is an associate editor of *IEEE Internet of Things Journal*, *IEEE Open Journal of Computer Society*, *IET Communications*, etc. He is the chair of the Multimedia Services and Applications over Emerging Networks Interest Group (MENIG) of the IEEE Comsoc Society, the Multimedia Communications Technical Committee. His research interests include wireless networking, mobile computing, and network security. He received the best paper award of IEEE ICC 2020, IEEE BigdataSE 2019, IEEE CyberSciTech 2017, etc.