Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 2568503, 17 pages
https://doi.org/10.1155/2022/2568503

Research Article

Prediction-Based Resource Deployment and Task Scheduling in Edge-Cloud Collaborative Computing

Mingfeng Su,[1,2] **Guojun Wang**,[3] **and Kim-Kwang Raymond Choo**[4]

[1]*School of Computer Science and Engineering, Central South University, Changsha 410083, China*
[2]*School of Business Information Technology, Hunan Vocational College of Commerce, Changsha 410205, China*
[3]*School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou 510006, China*
[4]*Department of Information Systems and Cyber Security, University Texas San Antonio, San Antonio, TX 78249, USA*

Correspondence should be addressed to Guojun Wang; csgjwang@gzhu.edu.cn

Received 13 September 2021; Revised 3 November 2021; Accepted 18 March 2022; Published 4 April 2022

Academic Editor: Yingjie Wang

Edge computing is becoming increasingly commonplace, as consumer devices become more computationally capable and network connectivity improves (e.g., due to 5G). With the rapid development of edge computing and Internet of Things (IoT), the use of edge-cloud collaborative computing to provide service-oriented network application (i.e., task) in edge-cloud IoT has become an important research topic. In this paper, we present an edge-cloud collaborative computing framework and our resource deployment algorithm with task prediction (RDAP). Based on our paradigm, tasks in the cloud service center are predicted using the two-dimensional time series, and task classification aggregation and delay threshold determination are combined to optimize task resource deployment of edge servers. A task scheduling algorithm with Pareto improvement (TSAP) is also proposed. At the edge servers, the Pareto progressive comparison is conducted in two stages to obtain the tangent point or any intersection point of the two objective curves of user's quality of service and effect of system service to optimize task scheduling. The experimental results show that for varying user task scales and different Zipf distribution α parameters, combining RDAP and TSAP (RDAP-TSAP) can improve the average user task hit rate. In addition, the average task completion time of users, the overall system service effect, and the total task delay rate of RDAP-TSAP are better than TSAP and the benchmark algorithms for task scheduling.

1. Introduction

As Internet of Things (IoT) and other related consumer devices (e.g., Internet of Vehicles and home/medical IoT) become more interconnected and pervasive in our digitally aware environment, there is a need for data analytics to be performed closer to the data sources [1–3]. Doing so allows us to improve users' quality of service, minimize latency, achieve privacy (to some extent), etc. This partly motivates network computing modes such as fog computing [4], transparent computing [5], edge computing [6, 7], and mobile edge computing [8].

In this paper, we focus on edge-cloud collaborative computing in edge-cloud IoT, to leverage the advantages of both cloud and edge computing for a range of services, such as data transmission, resources distribution, and service-oriented network application (i.e., task) offloading. By improving data collaborative processing of both cloud and edge servers, we can potentially minimize data processing delays, improve system scalability, achieve improved system services, etc. In addition, data may have spatiotemporal characteristics such as seasons [9–11], where there is periodic and trend information (explicit and implicit) in the spatiotemporal dimension. In other words, data changes or trends can be effectively predicted [12–14]. To implement edge-cloud collaborative computing in the edge-cloud IoT, cloud service center generally requires massive computing resources to perform data/predictive analytics, which can subsequently be used to guide the deployment of resources required for task operation of edge layer and promote efficient use of resources [15, 16]. At the edge layer, the edge server can balance the needs of users and service providers

through task collaborative offloading, and optimize task scheduling with multiple objectives, so as to enhance user service experience and improve the overall performance.

In edge-cloud IoT, the optimization of service caching (i.e., task resources) and task offloading (i.e., task scheduling) need to solve two problems. (1) The task load of different edge servers changes dynamically over time, and it is necessary to formulate a task resource push strategy according to task changes. (2) Task scheduling should not only consider the needs of users and shorten the task completion time, but also consider the interests of service providers and reduce the overall energy consumption of the system. Therefore, in edge-cloud IoT, we consider edge-cloud collaborative computing and introduce task prediction to study resource deployment and task scheduling optimization. The main contributions are as follows:

(i) A resource deployment algorithm with task prediction (RDAP) is proposed. In the cloud service center, task predictive analytics draw upon both horizontal and vertical time dimensions, and the deployment of resources required for the task operation of the edge servers is optimized to improve the average task hit rate (ATHR)

(ii) A task scheduling algorithm with Pareto improvement (TSAP) is proposed. At the edge server, considering the benefits of users and service providers, Pareto improvement is made to the two objectives of user's quality of service (QoS) and effect of system service (ESS) to optimize task scheduling

(iii) An edge-cloud collaborative computing framework is proposed, and an experimental environment for edge-cloud collaborative computing is constructed. Considering the impact of different user scales and Zipf distribution α parameters, the task scheduling is evaluated and analyzed from the average task completion time (ATCT), overall system service effect (OSSE), and total task delay rate (TTDR).

The rest of this paper is organized as follows. In Section 2, we introduce the related work. We present our designed framework and model of edge-cloud collaborative computing in Section 3. Then, in Sections 4 and 5, we, respectively, describe the task prediction and the resource deployment and task scheduling. Our evaluation setup and findings are presented in Section 6. Finally, we conclude this paper in Section 7.

2. Related Work

Task sharing on demand is one common application in our digitalized society [17, 18], and one of the key challenges is how to improve the QoS by reducing the task completion time. For example, Mao et al. [19] used reinforcement learning and neural network to design scheduling algorithm according to specific workloads, to efficiently schedule data processing jobs, and minimize the average job completion time in distributed clusters. Jalaparti et al. [20] proposed a scheduling method combining data placement and computational optimization to reduce cross rack data transmission and decrease task completion time. Ren et al. [21] predicted the stragglers in the cluster center, by leveraging the copy mechanism. Their approach also considers data location, task execution time, and task interdependence to reduce task delay in centralized and decentralized scheduling. To achieve increased ESS, cloud computing servers could attempt to enhance resource utilization and reduce system costs [22–24]. For example, Andrew et al. [25] put forward a new cluster scheduler for public IaaS platforms. Their scheduler is designed to dynamically allocate virtual machine (VM) instances to improve resource utilization and reduce costs. Liu et al. [26] proposed a joint execution strategy based on an improved genetic algorithm to reduce the overall energy consumption of the system. Nishtala et al. [27] designed a scalable QoS aware task management method, which uses deep reinforcement learning to reduce contention of shared resources, and minimizes data center energy consumption while ensuring QoS. These studies centrally execute user tasks in the cloud computing center. When task requests increase, coupled with reasons such as long transmission distances and limited backhaul links, problems such as high delay and increased energy consumption are likely to occur. The traditional cloud computing model is facing new challenges.

To mitigate some of the limitations associated with cloud computing, there have been attempts to utilize edge computing to offload all or part of the computationally intensive tasks to the edge servers and extend the computing power to the edge layer. Such an approach can potentially reduce data processing delays [28]. For example, Rodrigues et al. [29] proposed a method to minimize service delay, by offloading tasks that users cannot run to cloudlet servers at the edge network. They also attempt to reduce processing and transmission delays of tasks through VM migration and transmission power control. Mao et al. [30] presented a Lyapunov optimization-based dynamic computation offloading algorithm, which focuses on minimizing execution delay and execution cost of task failure, maximizing battery capacity of mobile devices, and gradually optimizing computationally intensive workloads to improve QoS and user's quality of experience (QoE). Chen et al. [31] used a mixed-integer nonlinear programming method to optimize task dispatch and resource allocation, solve mobile edge computing ultra-dense network task offloading, and minimize delays under the premise of considering device battery life. He et al. [32] proposed an incentive mechanism for online auction, which offloads user tasks to neighboring mobile devices to meet low latency requirements. However, these studies mainly focus on user's service needs and reduce the task completion time to optimize task offloading. Task offloading lacks consideration of system energy consumption optimization for service providers.

To enhance both QoS and ESS, edge computing needs to think of task offloading, network load, resource allocation, and transmission delay [33–35]. Wang et al. [36] proposed a local optimization algorithm for the univariate search technology, which introduces dynamic voltage scaling

technology in computational offloading, and uses variable replacement technology to find the optimal solution to minimize mobile energy consumption and application execution delay. Dinh et al. [37] put forward a task offloading framework from mobile devices to multiple edge devices, considering both fixed and flexible mobile device CPU frequencies based on the semi-definite relaxation approximation method to enhance task execution delay and device energy consumption. Zhang et al. [38] raised an energy-aware computing offloading scheme, which optimizes communication resources and computing resources allocation under limited energy and delay conditions, and finds out the hybrid nonlinear integer optimal solution of computing offloading and resource allocation through an iterative search algorithm. Wang et al. [39] came up with an optimal resource allocation scheme that combines AP energy transmission consumption, CPU processing frequency, user offloading file size, and user time allocation. The scheme thinks of computing and wireless power transmission to minimize the total AP energy consumption under the constraint of individual computing delay. Ding et al. [40] proposed a decentralized offloading strategy in a mobile edge computing environment with limited user equipment resources. The task execution location, CPU frequency, and transmission power are optimized based on code partition offloading to minimize application execution time and energy consumption. The above research work mainly considers task offloading. It is assumed that the edge server already has the relevant service cache required to execute the task, and service cache is also called task resource. When tasks are offloaded to edge servers, only the computing resource constraints are considered, but task resource constraints are not considered. However, in practical applications, the storage resources of edge servers are limited, and it is difficult to cache all task resources required for task execution [41]. It is necessary to dynamically formulate task resource deployment strategy according to the actual situation, and jointly optimize task resource and task offloading.

Therefore, the cloud computing capability is extended to the edge servers in edge-cloud IoT, and the edge-cloud collaborative computing framework is proposed. The task prediction is introduced to study the resource deployment and task scheduling optimization in the edge-cloud collaborative computing environment, and improve both QoS and ESS for users and service providers.

3. Framework and Modeling of Edge-Cloud Collaborative Computing

In this section, we first design the edge-cloud collaborative computing framework, and then model the edge-cloud collaborative computing, including quantifying user's quality of service and effect of system service. The main symbols and descriptions of this paper are shown in Table 1.

3.1. Edge-Cloud Collaborative Computing Framework. The edge-cloud collaborative computing framework is divided into the cloud layer, edge layer, and sensor layer, which are interconnected through the Internet. As shown in Figure 1,

Table 1: Description of main symbols.

Symbols	Description
U	Set of user
J	Set of task
E	Set of edge server
c	Cloud service center
q_u^j	QoS's coefficient of user u task j
r_u^j	System service revenue coefficient of user u task j
$\omega_{u,e}^j$	Proportion of task performed on receiving edge server for task j
$\omega_{e,e}^j$	Proportion of task assigned by the receiving edge server to other edge servers for task j
$\omega_{e,c}^j$	Proportion of task assigned by the receiving edge server to cloud service center for task j
f_e^j	Binary variable, 1/0 indicates whether the edge server e can execute the task j or not
l_u^j	The size of the task j for the end user u
k_u^j	Binary variable, 1/0 indicates whether the user u has the request task j or not
D	Distance coefficient between nodes
$d(x, y)$	Distance degree between nodes x and y
μ	Task execution preference weight coefficient
β	Revenue index
δ	Computing resources
ε	Storage resources
Δ	Task type, binary variable, 1/0 refers to time-sensitive and time-insensitive tasks, respectively
τ	Task execution time equivalent
∂	Average server energy consumption coefficient
ξ	Task delay occurrence threshold

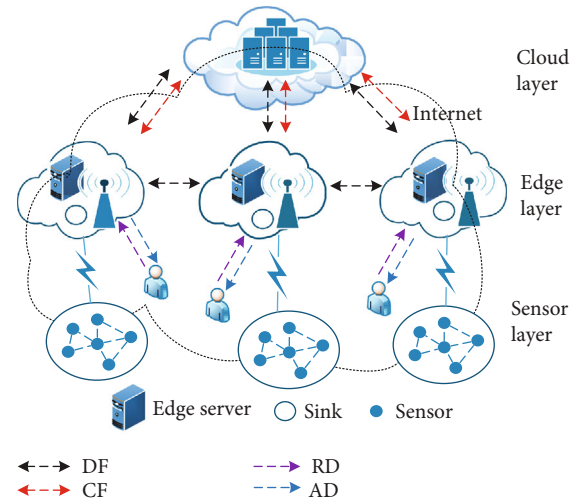


Figure 1: Edge-cloud collaborative computing framework.

edge-cloud collaborative computing can be applied in scenarios such as smart transportation, telemedicine, and environmental monitoring. The cloud layer includes a cloud service center, which is composed of some hardware such as homogeneous or heterogeneous computing, storage, network, and other hardware. The cloud service center uses virtualization, software defined network, redundancy, and other technologies to supply high-performance, highly reliable, and scalable resources to support a wide range of on-demand services for users. Control flow (CF) is generated between the cloud layer and edge layer. The cloud service center predicts the user task, and pushes the task resources to the edge servers through the CF in advance according to the prediction result. Task resources include software and software-dependent data required for task operation. The cloud service center monitors the task processing and resource utilization of the edge servers in real-time, summarizes the task processing and resource utilization of each edge server, and then distributes them to other edge servers through the CF.

The edge layer is composed of edge servers with limited resources, road side units (RSU), and sinks. The edge layer collects data from the sensor network in real-time. The edge server can aggregate and process data and provide users with real-time, fast, diverse, and flexible network applications. According to the current resource utilization and scheduling strategy, the edge server decides to execute the user tasks separately in the local edge servers, or subdivide and schedule the tasks to other edge servers and/or cloud service center for collaborative execution. Data flow (DF) is generated by executing tasks between edge-edge and edge-cloud. The edge server receives and loads the task resources from the cloud service center through the CF, and pre-start the environment required for task operation. The edge server uploads the task processing status and the resource usage status of computing, storage, and network to the cloud service center through the CF.

The sensor layer includes a sensor network, which is composed of a large number of sensors of IoT. Sensors can be deployed in environments such as smart transportation, telemedicine, and environmental monitoring. The sensor collects various required data in real-time and masters the status information of the monitoring area (object). The data collected by the sensors are uploaded to the edge layer in time. The uploaded data at the sensor layer can be analyzed and processed by the edge layer and cloud layer.

In edge-cloud IoT, users can initiate service-oriented task requests through mobile devices, computers, connected cars, and smart terminals, and send request data (RD) to the edge server. The requested task is executed by the local edge server alone or in collaboration with other edge servers and/or cloud service center, answer data (AD) will be returned to the client from the local edge server, other edge servers, and cloud service center. Take the application of edge-cloud collaborative computing in smart transportation as an example. Sensors in each area monitor passing vehicles and control traffic information, and upload the traffic information to the edge servers in each area in time. The edge server processes the data and uploads the summarized traffic information to the traffic command center (cloud service center). When a user initiates a task request, for example, the task is to obtain a regional traffic condition map. The edge server closest to the user receives the user's request. The edge server can perform tasks by itself, or dispatch to other adjacent edge servers and/or cloud service center to perform tasks cooperatively. Finally, the traffic condition map of a certain area is obtained, and the task result is returned to the user. The execution of user tasks requires the cloud service center to predict user tasks and push the task resources required to perform tasks to the edge server in advance. Task resources include software (program for drawing traffic conditions) and software data dependency (basic map).

3.2. Edge-Cloud Collaborative Computing Modeling.

$$q_u^j = l_u^j \left(\mu_u^e \cdot \omega_{u,e}^j / D_u^e + \mu_e^e \cdot \omega_{e,e}^j / D_{u,e}^e + \mu_e^c \cdot \omega_{e,c}^j / D_{u,e}^c\right). \quad (1)$$

Definition 1. Edge-cloud collaborative computing model (EC3M). EC3M is a six-tuple model, denoted as M_{EC3}, $M_{\mathrm{EC3}} = (U, J, E, c, O, \theta)$. U is the user set, which is composed of $n(u)$ independent users, $U = \{u_0, u_1, \cdots, u_{n(u)-1}\}$. Users do not interfere with each other, and the various tasks submitted by users have a time-series correlation, so the number and types of tasks can be predicted. J is the task set, which is composed of $n(j)$ service-oriented network applications (i.e., tasks), $J = \{j_0, j_1, \cdots, j_{n(j)-1}\}$. The task j is expressed as $j = \{\delta_j, \varepsilon_j, \Delta_j\}$. δ_j is the computing resources required for task j execution, which is quantified as the CPU computing power required for each task, i.e., GHz/task. ε_j is the storage resources required for task j execution. Δ_j is the task type of task j. $\Delta_j = 1$ indicates that task j is a time-sensitive task, and $\Delta_j = 0$ indicates that task j is a time-insensitive task. Each task type includes a variety of different tasks to meet the needs of various users. The task can be subdivided into several subtasks. E is the edge server set, which contains $n(e)$ geographically dispersed edge servers, $E = \{e_0, e_1, \cdots, e_{n(e)-1}\}$. The edge server $e = \{\delta_e, \varepsilon_e\}$ has limited hardware resources. δ_e and ε_e represent the computing resources and storage resources of the edge server e, respectively. The edge server is limited by hardware resources and can only load task resources required for some tasks simultaneously. The edge servers can subdivide the tasks and execute tasks locally or dispatch them to remote execution based on scheduling decisions. c is the cloud service center, which has massive computing, storage, network, and other hardware resources, and can load and run task resources for all tasks. The cloud service center manages and monitors the edge servers, effectively predicts the user tasks, and pushes the appropriate task resources to the relevant edge servers. O is the optimization objective of edge-cloud collaborative computing, which is quantified by both QoS and ESS (denoted as Q and S), $O = \{\max(Q), \max(S)\}$. θ is the optimization algorithm for resource deployment and task scheduling.

Definition 2. User's quality of service (QoS). QoS mainly focuses on the service experience and quality of user in the

edge-cloud collaborative computing environment. User tasks are executed locally in the edge servers that receive the tasks and the shorter the response time of user task requests, the higher the QoS. The latter's coefficient is stated in (1).

The QoS's coefficient is related to the task size and task execution. The task size is quantified as task execution time. $w^j_{u,e}$, $w^j_{e,e}$, and $w^j_{e,c}$ represent the proportion of task j executed locally by the receiving edge server, dispatched by the receiving edge server to other edge servers, and dispatched by the receiving edge server to cloud service center, respectively. Their corresponding task execution preference weight coefficients are μ^e_u, μ^e_e, μ^c_e, and there is $\mu^e_u > \mu^e_e > \mu^c_e > 0$, $\mu^e_u + \mu^e_e + \mu^c_e = 1$. The distance coefficient between nodes considers the sending and receiving of tasks and it is related to the distance degree between nodes. D^e_u is the distance coefficient between nodes that the task locally executed at the receiving edge server, $D^e_u = d(u, e) + d(e, u)$. $D^e_{u,e}$ is the distance coefficient between nodes that the task is dispatched by the local edge server to other edge servers, $D^e_{u,e} = d(u, e) + d(e, e) + d(e, u)$. $D^c_{u,e}$ is the distance coefficient between nodes that the task is dispatched by the local edge server to the cloud service center, $D^c_{u,e} = d(u, e) + d(e, c) + d(c, u)$. The distance degree $d(x, y)$ is related to the minimum bandwidth, cumulative delay, and reliability of the links between nodes x and y. The value of distance degree is equal to the cumulative delay divided by product of reliability and minimum bandwidth. If the minimum bandwidth is larger, the cumulative delay is smaller, and the reliability is higher, then the distance degree value is smaller. The objective function of QoS is described in (2).

$$Q = \sum_{u=0}^{u(n)-1} \sum_{j=0}^{j(n)-1} k^j_u \left(f^j_e \cdot q^j_u + \left(1 - f^j_e\right) \cdot \mu_0 \cdot l^j_u \cdot D^c_u \right) \cdot \begin{cases} \max\ (Q) \\ \text{s.t.} f^j_e = \{0, 1\}, \forall e \in E, j \in J\ . \\ k^j_u = \{0, 1\}, \forall u \in U, j \in J \end{cases} \tag{2}$$

$k^j_u = 1$ indicates that user u has a request for task j. $f^j_e = 1$ implies that the edge server e has the task resources required for task j to run. D^c_u is the distance coefficient between nodes of the task submitted by the user to the cloud service center, $D^c_u = d(u, c) + d(c, u)$. μ_0 is the non-preference weight coefficient, $\mu_0 = -\mu_e$. It can be seen from (2) that the edge servers have the resources required for the task operation and the larger the proportion of tasks executed locally, the larger the Q value, that is, the higher the user service quality of edge-cloud collaborative computing.

$$r^j_u = l^j_u \left(\mu^e_u \cdot \omega^j_{u,e} \cdot \beta^e_u + \mu^e_e \cdot \omega^j_{e,e} \cdot \beta^e_e + \mu^c_e \cdot \omega^j_{e,c} \cdot \beta^c_e \right). \tag{3}$$

Definition 3. Effect of system service (ESS). ESS mainly concentrates on the system service revenue and system service consumption of service providers in the edge-cloud collaborative computing. The system service revenue coefficient is stated in (3).

The system service revenue coefficient is associated with the task size and task revenue. β^e_u is the revenue index of the tasks performed on the local edge servers. β^e_e is the revenue index of the edge servers dispatching the received tasks to the neighbor edge servers. β^c_e is the revenue index of the edge servers dispatching the received tasks to the cloud service center. The objective function of ESS is shown in (4).

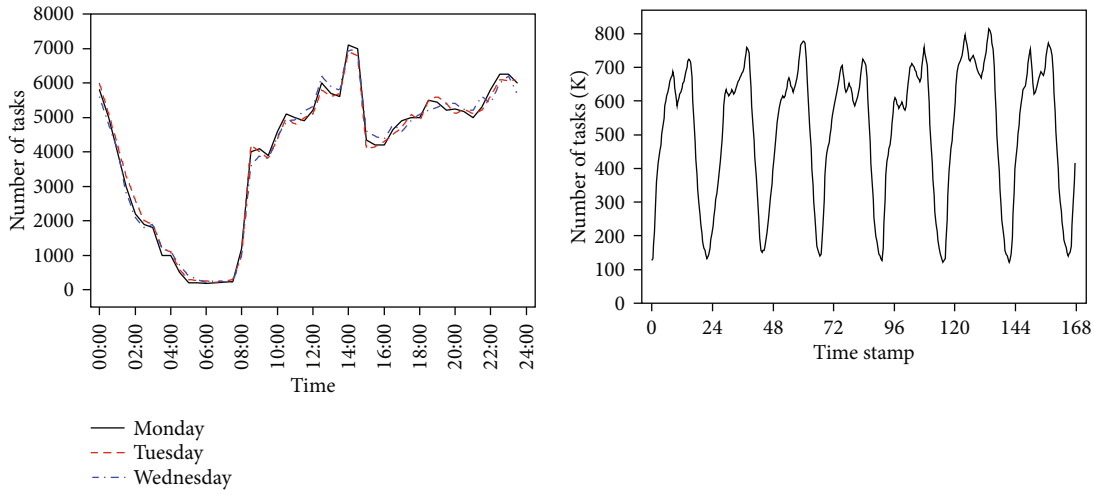
$$S = \sum_{u=0}^{u(n)-1} \sum_{j=0}^{j(n)-1} k^j_u \left(f^j_e \cdot r^j_u + \left(1 - f^j_e\right) \mu_0 \cdot l^j_u \cdot \beta^c_e \right) - \left(\tau_c \cdot \partial_c + \sum_{i=0}^{e(n)-1} \tau^i_e \cdot \partial^i_e \right) \begin{cases} \max\ (S) \\ \text{s.t.} f^j_e = \{0, 1\}, \forall e \in E, j \in J\ . \\ k^j_u = \{0, 1\}, \forall u \in U, j \in J \end{cases} \tag{4}$$

In (4), the system service consumption in the statistical period is the product of task execution time equivalent and average service energy consumption coefficient (denoted as τ and ∂). τ_c and τ_e represent the task execution time equivalent of the cloud service center and edge servers, respectively. ∂_c and ∂_e represent the average service energy consumption coefficient of cloud service center and edge servers, respectively. The energy consumption coefficient depends on the hardware/software costs, and the system operation and maintenance costs. The former includes the hardware/software purchase costs and the depreciation costs. The latter gets involved in the power consumption of equipment, the energy consumption of air conditioning and refrigeration, and the management and service costs. From the comparative analysis of single quantity, ∂_c is much larger than ∂_e. The higher the system service revenue and the lower the system service consumption, the larger the S value, that is, the higher the system service effect of edge-cloud collaborative computing.

In the edge-cloud collaborative computing model, the objective optimization of QoS and ESS is involved in resource deployment and task scheduling. (1) It needs to foretell the type and quantity of user tasks, reasonably push the task resources to the edge servers, and efficiently use the computing, storage, network, and other resources of the edge servers. (2) It also should optimize task scheduling, improve the QoS, and strengthen the ESS through the task collaborative processing of local edge servers, other edge servers, and cloud service center.

4. Task Prediction

Through long-term monitoring of user tasks, from a local perspective, the change of user tasks is a dynamic and random process, and a trend of change can be seen explicitly or implicitly. The task change and time have a strong correlation. From a global perspective, user tasks are



(a) External network interface of university data center

(b) Cluster server of e-commerce company

Figure 2: Task request statistics.

autocorrelated with time such as days, weeks, months, and years. Based on the task change rules of the horizontal and vertical time dimensions, the task changing trend can be found through prediction. Figure 2(a) shows the task request statistics chart of the network external network interface of a university's data center. Figure 2(b) shows the statistics of the number of tasks executed in the cluster server of an e-commerce platform. Both figures use hours as the statistical unit, and the number of tasks changes with time. By observing the number of tasks listed in Figure 2(a) for 3 consecutive days and Figure 2(b) for 7 consecutive days, it is found that the changing trend of daily tasks in the same period is similar, and the overall shows the regular periodic fluctuations. This provides a reference basis for user task prediction in edge-cloud IoT.

According to the overlapping trend of strong periodicity, medium trend, and weak randomness of user tasks in edge-cloud collaborative computing, the user tasks can be comprehensively predicted from the horizontal and vertical time dimensions in the cloud service center. Through statistics and analysis of the past time-series data, the change of tasks can be inferred. The prediction model based on the two-dimensional time series is indicated in (5).

$$p_t = \lambda v_t + (1-\lambda)\eta_t + z_t. \tag{5}$$

p_t is the task prediction result of the next time series. v_t shows the current time series value of the vertical time dimension, which is the periodic statistics part, specifically the mean value of the same time series in different periods. v_t is the mean value of the same time series in different periods. η_t is the current time series value of the horizontal time dimension, which is the trend prediction part. z_t is the random noise part. λ represents the adjustment factor of two-dimensional time series, and its value is between 0 and 1.

Theorem 4. *The task of trend prediction based on the horizontal time dimension in edge-cloud collaborative computing is* $\eta_t = (\eta_{t-1} + \eta_{t-2} + \cdots + \eta_{t-n}/n) + (n + 1/2)\varphi$.

Proof. According to the statistical analysis of practical data, the number of tasks increases or decreases linearly from a local perspective. The change in the number of tasks at time t can be expressed by a linear equation, as shown in (6). Thus, the number of tasks with time $t - n$ can be obtained, as shown in (7).

$$\eta_t = \varphi\chi_t + a, \tag{6}$$

$$\eta_{t-n} = \varphi\chi_{t-n} + a. \tag{7}$$

According to the moving average prediction method, the number of tasks at time t is related to the number of tasks in the previous n time slots, as shown in (8).

$$\bar{\bar{\eta}}_t = \frac{\eta_{t-1} + \eta_{t-2} + \cdots + \eta_{t-n}}{n}. \tag{8}$$

Combining formula (7), and calculating $\bar{\bar{\eta}}_t$ to obtain (9).

$$\begin{aligned}\bar{\bar{\eta}}_t &= \frac{\varphi\chi_{t-1} + a + \varphi\chi_{t-2} + a + \cdots + \varphi\chi_{t-n} + a}{n} \\ &= \frac{\varphi(\chi_{t-1} + \chi_{t-2} + \cdots + \chi_{t-n}) + na}{n} \\ &= \frac{\varphi(n\chi_t - n(n+1)/2) + na}{n} \\ &= \varphi\chi_t - \varphi(n+1)/2 + a = \varphi\chi_t + a - \varphi(n+1)/2.\end{aligned} \tag{9}$$

According to formula (6), the number of tasks $\eta_t = \varphi\chi_t + a$ at time t is substituted into formula (9), and (10) is obtained.

$$\bar{\bar{\eta}}_t = \eta_t - \varphi(n+1)/2. \tag{10}$$

There is a delay deviation of $\varphi(n+1)/2$ between $\bar{\bar{\eta}}_t$ (the number of tasks predicted by the moving average method) and η_t (the actual number of tasks), which needs to be corrected. Therefore, the task of trend prediction based on the horizontal time dimension is $\eta_t = \bar{\bar{\eta}}_t + \varphi(n+1)/2$, which is (11). The φ value can be calculated by a linear regression formula, $\varphi = \sum \chi_i \sum \eta_i - n\bar{\chi}\bar{\eta} / \sum \chi_i^2 - n\bar{\chi}^2$. It should be pointed out that if the number of tasks is absolutely stationary in the statistical interval, its $\varphi = 0$.

$$\eta_t = \frac{\eta_{t-1} + \eta_{t-2} + \cdots + \eta_{t-n}}{n} + \frac{n+1}{2}\varphi \tag{11}$$

□

Thus, the task prediction based on the two-dimensional time series is gotten, as stated in (12).

$$p_t = \lambda v_t + (1-\lambda)\left(\frac{\eta_{t-1} + \eta_{t-2} + \cdots + \eta_{t-n}}{n} + \frac{\eta+1}{2}\varphi\right) + z_t. \tag{12}$$

5. Resource Deployment and Task Scheduling

In this section, we introduce the resource deployment and task scheduling optimization of edge-cloud collaborative computing. Firstly, the cloud service center pushes the task resources to the appropriate edge servers according to the task prediction result to improve the local execution rate of user tasks. Secondly, the edge servers optimize the task scheduling through Pareto improvement to enhance both QoS and ESS.

5.1. Resource Deployment Algorithm with Task Prediction (RDAP). In the edge-cloud collaborative computing environment, the hardware resources of the edge servers are limited, and the tasks they perform are also limited. The cloud service center needs to monitor the task processing status of the edge layer in real-time, use task prediction based on two-dimensional time series to obtain the changing trend of task type and quantity, and push the task resources to the edge servers. Task running needs to occupy hardware resources, this paper mainly considers computing resources (δ) and storage resources (ε) [42]. The maximum available resource of the edge server e is h_e, $h_e = \{\delta_e^{\max}, \varepsilon_e^{\max}\}$. The currently available resource of the edge server e is b_e, $b_e = \{\delta_e^{\text{cur}}, \varepsilon_e^{\text{cur}}\}$. The resource consumption of task j is $g_j = \{\delta_j, \varepsilon_j\}$. Considering the limited number of tasks performed by the edge servers, the cloud service center needs to classify and aggregate the prediction tasks, and control the number of task resources pushed to the edge servers, keeping the task load of the edge servers in a reasonable range. The RDAP is shown in Algorithm 1.

The resource deployment with task prediction is performed in the cloud service center. Tasks $j \in J$ are classified into two types: time-sensitive task ($\Delta_j = 1$) and time-insensitive task ($\Delta_j = 0$) (Line 1). According to the monitoring data, the current and maximum available resources of each edge server are calculated (Lines 2-3). Formula (12) is used to predict tasks based on the two-dimensional time series (Line 4). The tasks of each edge server are classified and aggregated in line with the prediction result, arranging them in descending order by the occurrence frequency (Line 5). This can reduce the push quantity of task resources, decrease the resource occupation of the edge servers, and improve the hit rate of user tasks under the resource shortage. Based on the task classification and aggregation result, the resource deployment of time-sensitive tasks is first considered, and then the resource deployment of time-insensitive tasks is taken into account when the edge servers have surplus resources available, which is conducive to ensuring the user's quality of service (Lines 6-23). After obtaining the task resources deployed by each edge server, the currently available resources of each edge server are updated. To decrease the task delay, the delay threshold is determined to make sure that the proportion of available resources (including computing resources and storage resources) of the edge servers is higher than the task delay occurrence threshold ξ, so as to control the number of task resources pushed to the edge servers. Finally, the edge server task resource deployment set X is returned, and the cloud service center pushes the task resources to the edge servers (Line 25). The time complexity of the RDAP is $O(n(e) \times n(j)(\text{lb}(n(j)) + 1))$.

5.2. Task Scheduling Algorithm with Pareto Improvement (TSAP). The task scheduling of the edge-cloud collaborative computing should be oriented to users and service providers, and the QoS and ESS ought to be considered comprehensively. The single objective optimization of task scheduling cannot ensure that the other objective is also optimal. It is necessary to weigh two objectives for comprehensive optimization. For example, task scheduling to nodes with strong computing capabilities can reduce the response time of tasks and improve QoS, but it is easy to increase system service consumption and reduce ESS. It is necessary to consider the interests of users and service operators to weigh QoS and ESS. Therefore, the Pareto improvement in the field of economics is introduced, and the task scheduling scheme of edge-cloud collaborative computing is obtained by seeking Pareto improvement for both QoS and ESS. The TSAP is described in Algorithm 2.

The task scheduling with Pareto improvement is performed in the edge servers. New tasks are received and added to the task set J. If multiple users initiate task requests at the same time period, the tasks are sorted in ascending order (Lines 1-6). Pareto improvement of task scheduling is involved two stages (Lines 7-21). The first stage is the objective optimization of the QoS (Lines 8-13). The m-group of task scheduling schemes in the first stage is solved based on the random greedy approximation algorithm. Formula (2) is used to calculate the Q value in each group, and select the scheduling scheme with the highest Q value in each group. The objective curve of QoS can be obtained from these schemes set. The second stage is the objective optimization of the ESS (Lines 14-19). Based on

Input: E: edge server set; J: task set; H: edge server maximum available resource set; B: edge server current available resource set.
Output: X: edge server task resource deployment set.

classify all j into time-sensitive or time-insensitive task, $\exists j \in J, \Delta_j = \{1, 0\}$;
for each $e \in E$ **do**
  calculate h_e and b_e, $\exists h_e = \{\delta_e^{\max}, \varepsilon_e^{\max}\}$, $b_e = \{\delta_e^{\text{cur}}, \varepsilon_e^{\text{cur}}\}$;
  predict p_t by formula (12);
  sort, aggregate, and rank tasks for J in descending order of frequency;
  for each $j \in J$ **do**
    if $f_e^j = 1 \cap \Delta_j = 1$ **then**
      if $\min\{(\delta_e^{\text{cur}} - \delta_j/\delta_e^{\max}), (\varepsilon_e^{\text{cur}} - \varepsilon_j/\varepsilon_e^{\max})\} > \xi$ **then**
        $\delta_e^{\text{cur}} \longleftarrow \delta_e^{\text{cur}} - \delta_j$, $\varepsilon_e^{\text{cur}} \longleftarrow \varepsilon_e^{\text{cur}} - \varepsilon_j$;
        update $X \longleftarrow X_e^j$;
      end if
    end if
  end for
  if $\min\{\delta_e^{\text{cur}}/\delta_e^{\max}, \varepsilon_e^{\text{cur}}/\varepsilon_e^{\max}\} > \xi$ **then**
    for each $j \in J$ **do**
      if $f_e^j = 1 \cap \Delta_j = 0$ **then**
        if $\min\{(\delta_e^{\text{cur}} - \delta_j/\delta_e^{\max}), (\varepsilon_e^{\text{cur}} - \varepsilon_j/\varepsilon_e^{\max})\} > \xi$ **then**
          $\delta_e^{\text{cur}} \longleftarrow \delta_e^{\text{cur}} - \delta_j$, $\varepsilon_e^{\text{cur}} \longleftarrow \varepsilon_e^{\text{cur}} - \varepsilon_j$;
          update $X \longleftarrow X_e^j$;
        end if
      end if
    end for
  end if
end for
return X

Algorithm 1: RDAP.

Input: E: edge server set; J: task set; c: cloud service center; μ_u^e, μ_e^e, μ_e^c; β_u^e, β_e^e, β_e^c; ∂_c, ∂_e.
Output: Y: task scheduling scheme set.

new j received
  $J \longleftarrow J + j$;
end if
if count$(J) \neq \varnothing$ **then**
  sort J in ascending order;
end if
for each $j \in J$ **do**
  for each $E \cup C$ **do**
    if $f_e^j = 1, \exists j \in J, e \in E$ **then**
      calculate Q according to formula (2);
      select $Y_j^1[m] \subseteq Y^1$, $Y_j^1[m] \succ \max(Q)$;
    end if
  end for
  for each $E \cup C$ **do**
    if $f_e^j = 1, \exists j \in J, e \in E$ **then**
      calculate E according to formula (4);
      select $Y_j^2[m] \subseteq Y^2$, $Y_j^2[m] \succ \max(S)$;
    end if
  end for
  $Y \longleftarrow Y_j^1[m] \cap Y_j^2[m]$;
end for
return Y

Algorithm 2: TSAP.

the random greedy approximation algorithm, the m-group task scheduling schemes in the second stage are solved. The S value in each group is calculated by using formula (4), and the scheduling scheme with the highest S value in each group is selected. The objective curve of ESS can be obtained by these schemes set. The task scheduling schemes obtained in two stages are gradually compared by Pareto improvement, and the tangent point or any intersection point corresponding to the QoS and ESS objective curves are selected. The tangent point is the only optimal scheme for the two objectives, and the intersection point is any co-optimal scheme for the two objectives, so as to obtain the optimal scheduling scheme for each task (Line 20). Finally, the task scheduling scheme set Y is returned (Line 22). The time complexity of the TSAP is $O(n(j) \times (\mathrm{lb}(n(j)) + (n(e) + 1) \times m))$.

6. Experimental Evaluation

In this section, we build an edge-cloud collaborative computing experimental environment, set experimental parameters, determine evaluation indexes, and comprehensively evaluate resource deployment and task scheduling.

6.1. Experimental Environment and Parameters. The hardware platform of the edge-cloud collaborative computing experiment is an x86 server with Intel e5-2620v4 CPU, 64GB ECC RAM, and 3 × 2TB STA hard disk. The software platform relies on the CentOS 8.0 x86_64 operating system, and uses python 3.8 to build an edge-cloud collaborative computing simulation environment. In addition, OriginPro 2017 software is used for data analysis and post drawing. The parameters of the experimental environment are shown in Table 2. In the edge-cloud collaborative computing environment, the user task type, task execution time, CPU resource utilization, and RAM resource utilization are set by referring to the data set published by Alibaba Cloud [42–44]. User tasks have time series correlation, and the overall distribution is Zipf [45–47]. The frequency of tasks is inversely proportional to the rank of task popularity. In the experiment, the default value of Zipf distribution α parameter is 1.0 [48, 49]. The maximum available resources of the cloud service center are 10 K PE CPU, 10 TB RAM, and 10 PB disk. The hardware resource parameters of the edge servers are shown in Table 3. The benchmark algorithms for task scheduling include BAO (the benchmark task scheduling algorithm with OREO) [50] and BAF (the benchmark task scheduling algorithm with FIFO) [51, 52]. Task scheduling evaluation metrics include ATCT, OSSE, and TTDR.

6.2. Experimental Results and Discussion

6.2.1. Resource Deployment. In the edge-cloud collaborative computing environment, resource deployment with task prediction is evaluated and analyzed. The day is taken as the statistical cycle, each cycle is divided into 24 time periods, and each time period is 0.5 hours. In the experiment, the cloud service center predicts the tasks of the edge servers based on the two-dimensional time series, and the value of time slot n is 10. According to the experimental data

Table 2: Experimental environment parameters.

Parameters	Value
Types of tasks	300
Number of edge servers	15
δ_j/GHz	[0.1-0.3]
ε_j/MB	[20-80]
δ_e/GHz	[2.5-3.6]
μ_u^e, μ_e^e, μ_e^c	0.48, 0.31, 0.21
β_u^e, β_e^e, β_e^c	0.39, 0.36, 0.25
ξ	0.20

Table 3: Hardware resource parameters of edge servers.

Edge server number	CPU/PE	RAM/GB	Disk/GB
E01-E03	2 ~ 4	4 ~ 8	500
E04-E06	4 ~ 6	4 ~ 8	500
E07-E09	4 ~ 6	8 ~ 16	1000
E10-E12	4 ~ 8	8 ~ 16	1000
E13-E15	6 ~ 16	16 ~ 32	1000

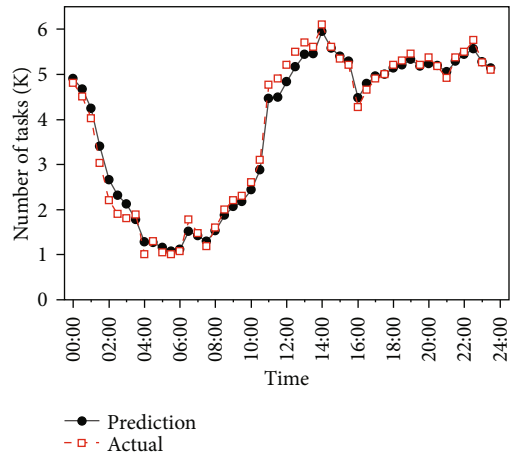


Figure 3: Prediction of user tasks.

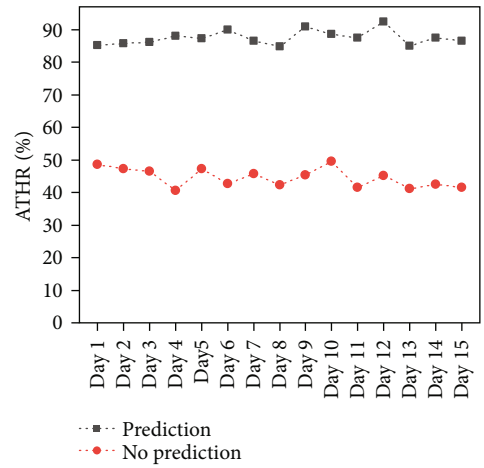


Figure 4: Average task hit rate of edge servers.

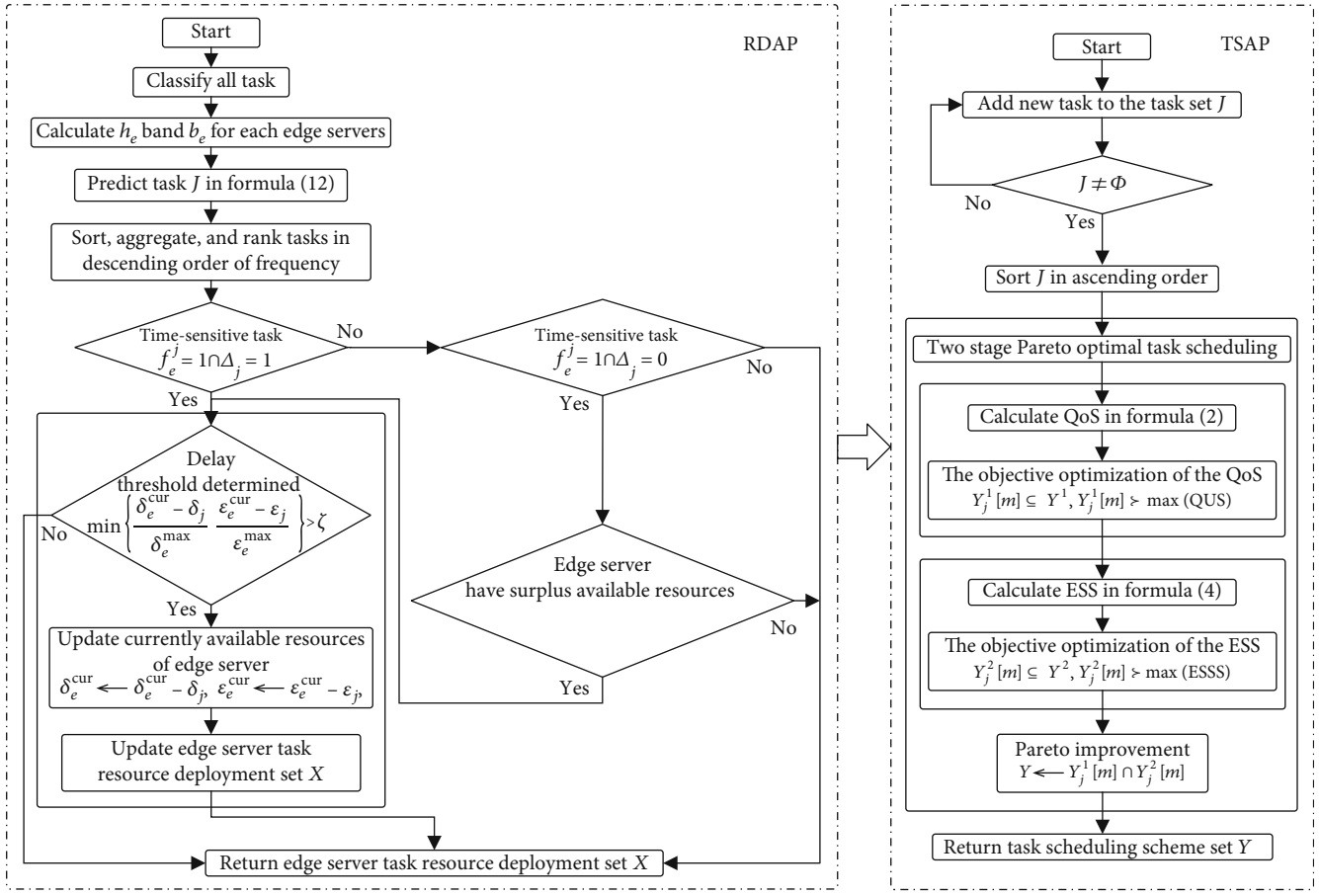


Figure 5: RDAP-TSAP.

of previous task prediction, the adjustment factor λ of the two-dimensional time series is set up to 0.27, and the random noise is ignored. The predicted result is compared with the actual number of tasks processed by the edge servers. As shown in Figure 3, the actual value of user tasks changes greatly, and the predicted value of tasks changes relatively gently. In general, the task prediction based on the two-dimensional time series is close to the actual value, the deviation of the average task number is less than 5%, and the accuracy of task prediction is high. The result can guide the cloud service center to optimize the task resource deployment of the edge servers.

The average task hit rate (ATHR) is used to evaluate the effect of resource deployment with task prediction. ATHR refers to the average value of the local execution proportion of tasks of each edge server. The proportion of local task execution of the edge server is the ratio of the edge server receiving user task requests and executing tasks locally (because it has the task resources required for task operation). The high ATHR value indicates that the cloud service center has a high accuracy rate of pushing the task resources based on the predicted result. The high rate of local execution of tasks in the edge servers is conducive to improving both QoS and ESS. In the edge-cloud collaborative computing environment, the cloud service center can effectively predict user tasks based on the two-dimensional time series, classify and aggregate them according to the prediction result, and optimize the task resources pushed to each edge server. It can improve the local execution rate of user tasks on the edge servers, and reduce the passive application of task resources to the cloud service center because the edge servers do not have the resources required for task operation. Through 15 consecutive days of experiments, using task prediction based on the two-dimensional time series to deploy the task resources required by the edge servers, the average task hit rate of the edge servers is very high. As shown in Figure 4, its ATHR value reaches 85.79% ~ 92.39%, which is much higher than the average task hit rate of less than 50% for resource deployment without task prediction.

6.2.2. Task Scheduling. We evaluate and analyze the task scheduling performance of the TSAP, RDAP-TSAP (combining RDAP and TSAP, as shown in Figure 5), BAO, and BAF from the three aspects of ATCT, OSSE, and TTDR. In the edge-cloud collaborative computing environment, to compare different task scheduling performances, the m-group task scheduling schemes of TSAP consider the values of 3, 5, 7, and 9, respectively.

(1) Average Task Completion Time (ATCT). The ATCT is the ratio of the total completion time of all user tasks to

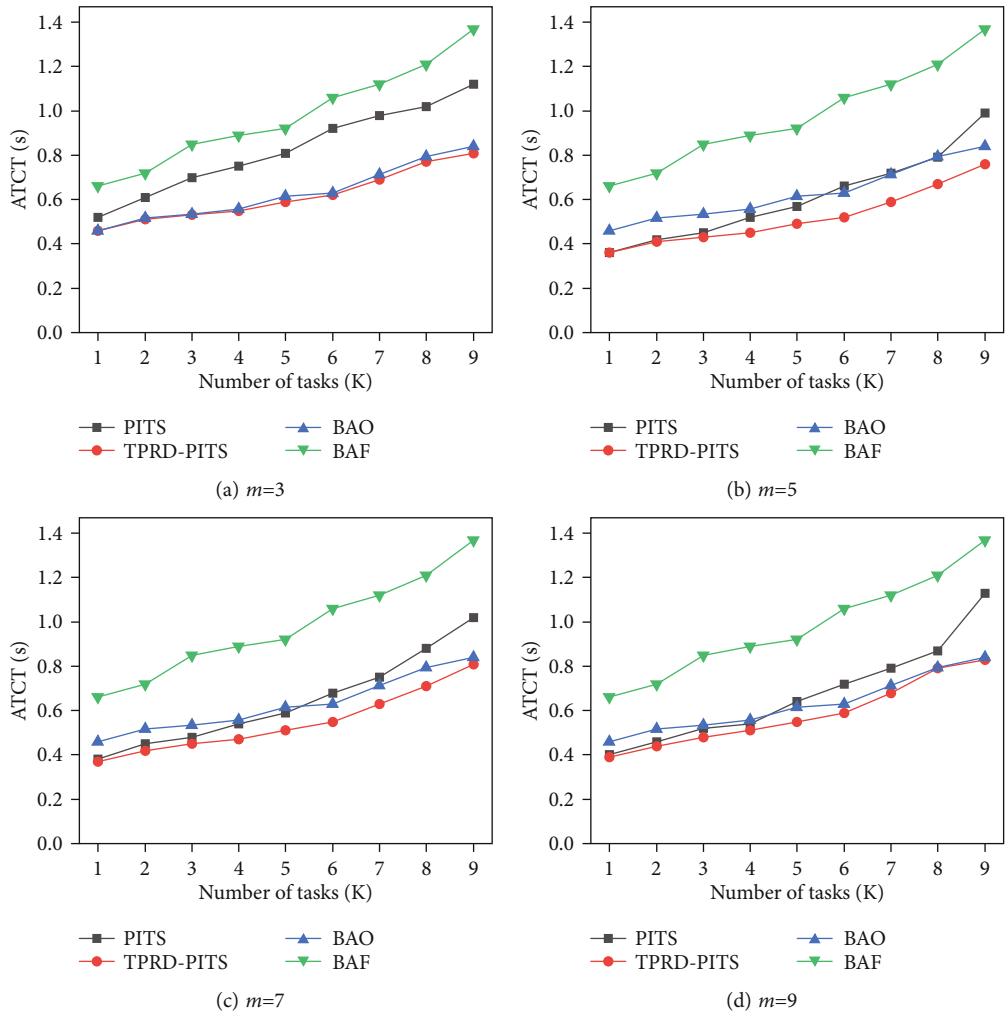


Figure 6: Average task completion time under different tasks.

the number of user tasks during the statistical period. The ATCT mainly considers the user service experience. The smaller the ATCT value, the shorter the average completion time of all tasks and the higher the quality of user experience. In the edge-cloud collaborative computing environment, the Zipf distribution α parameter of the user task is set to the default value, and the ATCT of the TSAP, RDAP-TSAP, BAO, and BAF algorithms are compared and analyzed by changing the task scale. As shown in Figure 6, with the increase of the number of user tasks, the ATCT values of the four algorithms increase and show an approximately linear trend. The ATCT values of RDAP-TSAP, BAO, and TSAP algorithms are significantly lower than that of BAF algorithm. The TSAP algorithm effectively reduces the ATCT by Pareto improvement on the two objectives of QoS and ESS, and its ATCT value is reduced by 29.69% on average compared with the BAF algorithm. The BAO algorithm jointly optimizes task caching and task offloading, and its ATCT value is reduced by 35.68% on average compared with the BAF algorithm. The RDAP-TSAP algorithm predicts the number of user task types, improves the accuracy of resources required by the cloud service center to push tasks to edge servers, and further reduces the ATCT by Pareto improvement of QoS and ESS. The ATCT value of RDAP-TSAP algorithm is 42.07% lower than the BAF algorithm, and 9.94% lower than the BAO algorithm. In addition, the TSAP and RDAP-TSAP algorithms have the best ATCT values when m =5. It shows that the smaller m value leads to the single task scheduling scheme, and the higher m value will increase the calculation cost of task scheduling itself, which will improve the ATCT.

Considering the influence of task Zipf distribution α parameter, the ATCT of the TSAP, RDAP-TSAP, BAO, and BAF algorithms will be further evaluated under certain user tasks. The number of user tasks is set up to 5k in the edge-cloud collaborative computing environment. As shown in Figure 7, as the value of the Zipf distribution α parameter increases, user tasks become more and more concentrated in the popular task types, and the ATCT values of the four scheduling algorithms decrease in varying degrees. Moreover, the TSAP and RDAP-TSAP algorithms have the best ATCT value when the m-group of task scheduling schemes is 5. Overall analysis, the RDAP-TSAP algorithm predicts the type and number of tasks, improves the accuracy of task resources push, increases the average task hit rate; and then integrates both QoS and ESS for Pareto improvement to

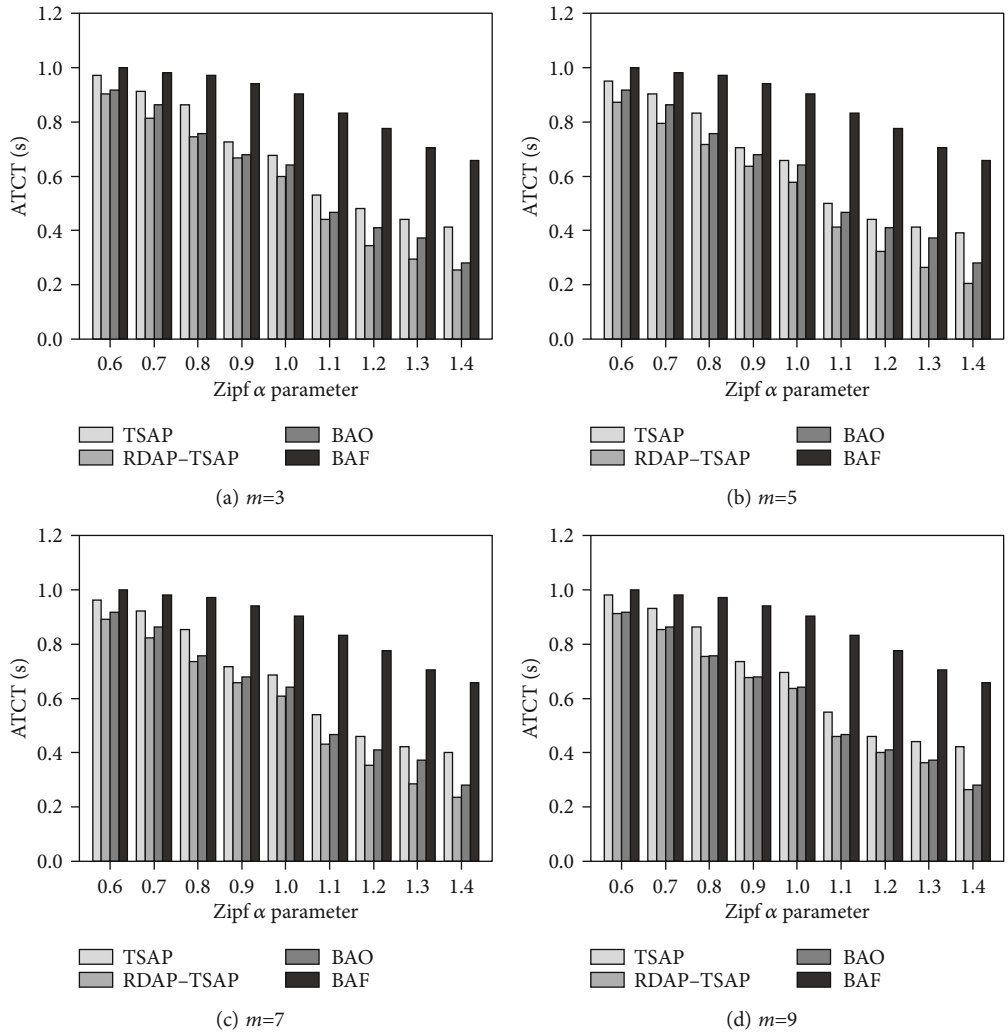


Figure 7: Average task completion time under different Zipf distribution α parameters.

optimize task scheduling. Its ATCT value is the best, with an average overall reduction of 34.94% compared with the BAF algorithm and 6.22% compared with the BAO algorithm. The TSAP algorithm benefits from two-stage Pareto improvement of QoS and ESS. In the absence of task prediction to optimize resource deployment, its ATCT value is overall reduced by 23.23% on average compared with the BAF algorithm.

(2) Overall System Service Effect (OSSE). The OSSE is the difference between system service revenue and system service consumption of the cloud service center and edge servers in the statistical period, which is the ESS value of normalization. The OSSE mainly considers the interests of service providers. The higher the system service revenue of providers and the lower their system service consumption, the higher its OSSE value. In the edge-cloud collaborative computing environment, the user tasks are Zipf distribution with default α parameter. The OSSE of the TSAP, RDAP-TSAP, BAO, and BAF algorithms are evaluated by gradually increasing the number of user tasks. As shown in Figure 8, the OSSE values of the four algorithms increase with the number of user tasks, showing an approximate logarithmic growth. Compared with the BA algorithm, the OSSE values of the RDAP-TSAP, BAO, and TSAP algorithms are significantly higher. Among them, the TSAP algorithm optimizes task scheduling through Pareto improvement of both QoS and ESS objectives, and improves the overall service effectiveness of the cloud service center and edge servers. Its OSSE value overall increases by 24.69% on average. The BAO algorithm jointly optimizes task caching and task offloading, and its OSSE value increases by 36.78% on average. The RDAP-TSAP algorithm predicts based on the type and number of tasks in the cloud service center, effectively pushes the task resources, improves the average task hit rate, and optimizes task scheduling based on the objectives of QoS and ESS at the edge servers to further increase the OSSE value. Its OSSE value overall increases by 49.55% on average. In addition, compared with the BAO algorithm, the OSSE value of the RDAP-TSAP algorithm has an overall increase of 10.43% on average. Moreover, similar to the previous ATCT analysis, the OSSE values of the TSAP and RDAP-TSAP algorithms are optimal when m=5. It indicates that a small value of m leads to a single task scheduling scheme

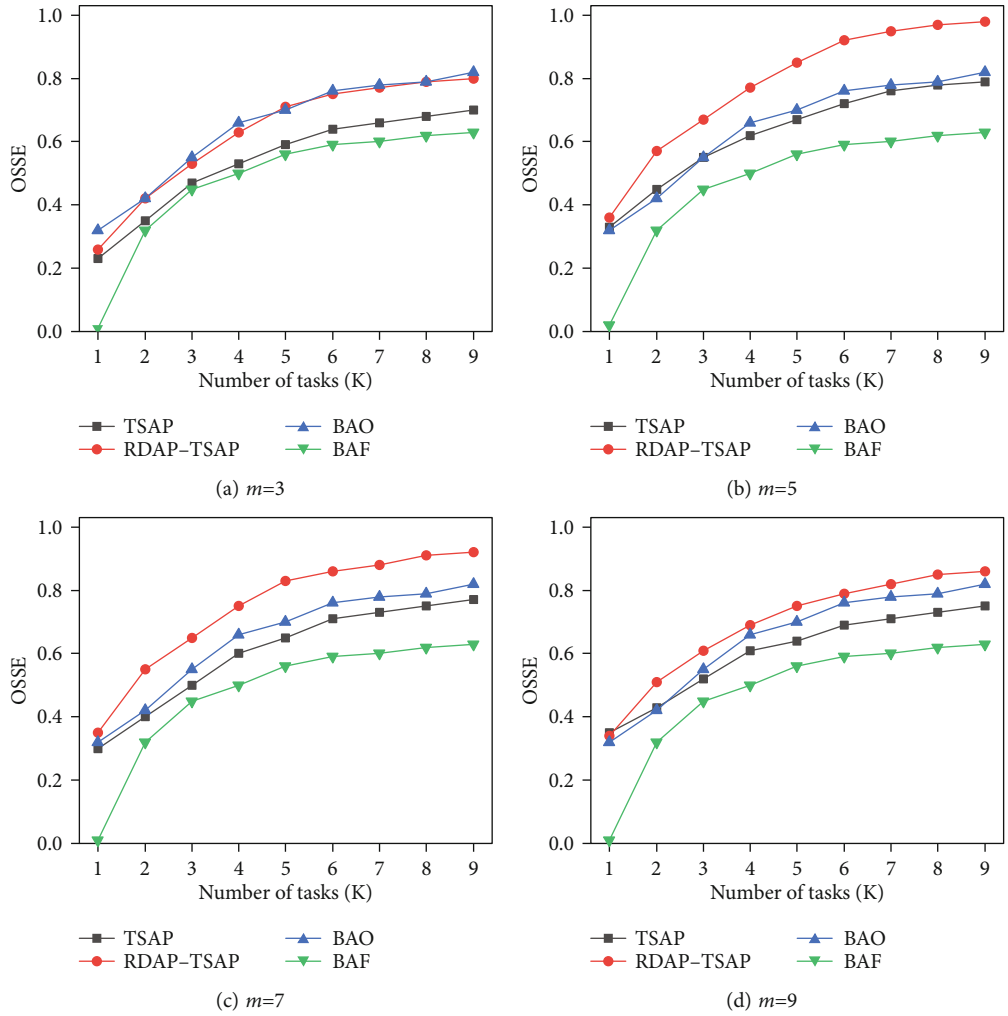


Figure 8: Overall system service effect under different tasks.

and is under-optimized, and a large value of m increases the computational overhead of task scheduling, which will reduce the OSSE.

Considering the influence of task Zipf distribution α parameter, the OSSE of the TSAP, RDAP-TSAP, BAO, and BAF algorithms will be further evaluated under certain user tasks. In Figure 9, the number of user tasks is 5k in the edge-cloud collaborative computing environment. As the value of the Zipf distribution α parameter increases, the user tasks gradually tend to be more popular task types and decrease the number of task resources required by edge servers, and the OSSE values of the four algorithms are greatly improved. And similar to the previous ATCT analysis, the OSSE values of the TSAP and RDAP-TSAP algorithms are optimal when the value of the m-group task scheduling schemes is 5. In general, the RDAP-TSAP algorithm predicts the tasks, improves the accuracy of task resources required to push edge servers, and increases the average task hit rate. Moreover, through the two-stage comprehensive optimization considering the QoS and ESS, the OSSE value of RDAP-TSAP algorithm is the best. Its OSSE value is 39.54% higher than the BAF algorithm, and 11.91% higher than the BAO algorithm. In the absence of task prediction, the TSAP algorithm uses Pareto optimization of the QoS and ESS, and its OSSE value is overall increased by 22.62% on average compared to the BAF algorithm.

(3) Total Task Delay Rate (TTDR). The TTDR is the proportion of tasks whose task response time exceeds 3 times the average response time of similar tasks in the statistical time period. The TTDR affects the user's quality of service and the effect of system service [53–55]. The low TTDR value can help reduce the average task completion time of users and enhance the QoS, and also help reduce the system energy consumption and improve the ESS. In the edge-cloud collaborative computing environment, user tasks are Zipf distribution with default α parameter, and the value of the m-group task scheduling schemes is 5. The TTDR of the TSAP, RDAP-TSAP, BAO, and BAF algorithms are evaluated by gradually increasing the number of user tasks. As shown in Figure 10, with the number of user tasks increasing, the TTDR values of the four algorithms increase. In general, the RDAP-TSAP algorithm is the best, the BAO and TSAP algorithm is the second, and the BAF algorithm is the worst. The RDAP-TSAP algorithm predicts tasks, classifies and aggregates tasks to reduce the number of task

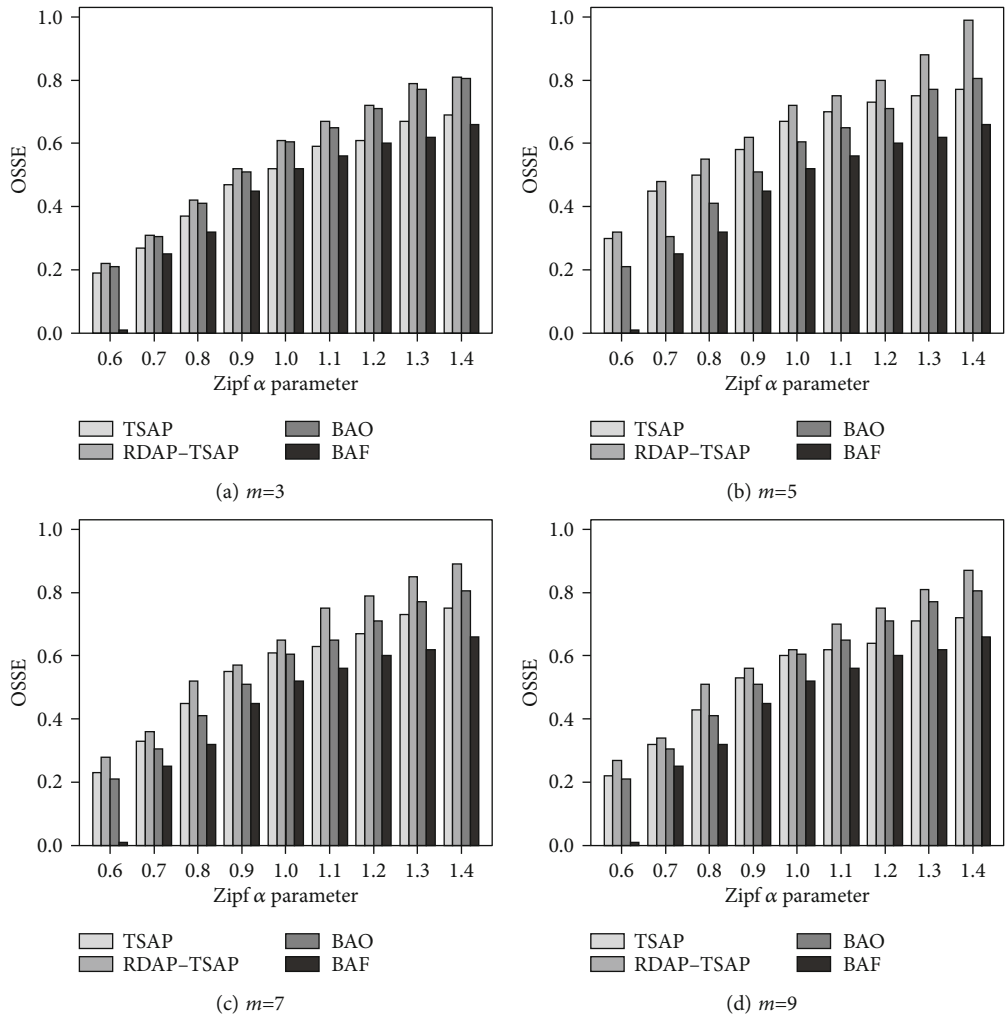


Figure 9: Overall system service effect under different Zipf distribution α parameters.

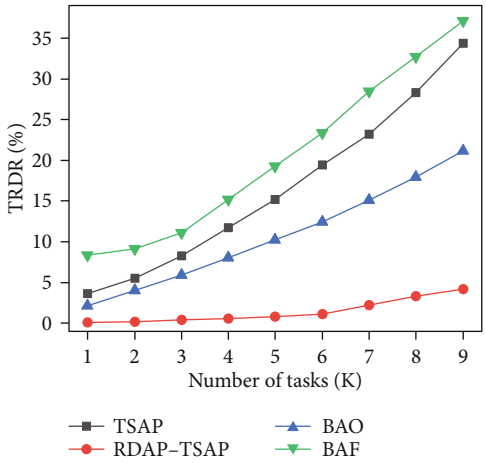


Figure 10: Total task delay rate under different tasks.

resources pushed to the edge servers, determines the delay threshold during resource deployment, and considers the available resources of the edge servers to reduce the probability of task delay. As the number of tasks increases, the total task delay rate increases slowly, and the TTDR value is controlled between 0.12% and 4.17%. The BAF, TSAP, and BAO algorithms lack task prediction in task scheduling and do not handle possible task delays. Their TTDR values are high, and the total task delay rate increases linearly with the increase of tasks. The BAF algorithm has the highest TTDR value, ranging from 8.37% to 37.12%. Because the TSAP algorithm performs Pareto improvement for task scheduling by integrating the objectives of QoS and ESS, it can reduce the average service completion time of tasks, and its TTDR value is relatively low, ranging from 3.67% to 34.39%. The BAO algorithm jointly optimizes task caching and task offloading, and its TTDR value is 2.17% to 21.19%.

Considering the influence of task Zipf distribution α parameter, the TTDR of the TSAP, RDAP-TSAP, BAO, and BAF algorithms is further evaluated under the condition of unchanged user tasks. In the edge-cloud collaborative computing environment, the number of user tasks is set up to 5k, and the value of the m-group task scheduling schemes is 5. The changes in their TTDR are shown in Figure 11. As the value of Zipf distribution α parameter increases, user

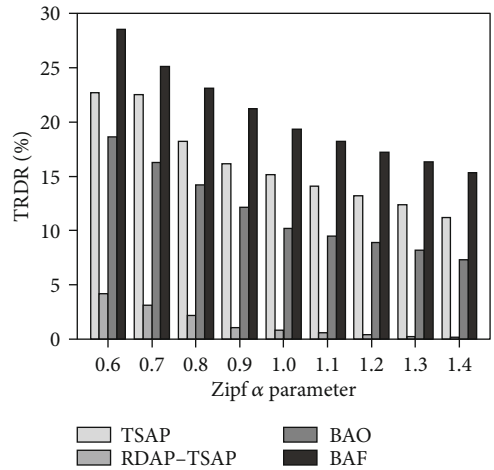


Figure 11: Total task delay rate under different Zipf distribution α parameters.

tasks are more and more concentrated on task types with high popularity, and the task resource types required by edge servers are reduced. In general, the TTDR of the TSAP, RDAP-TSAP, BAO, and BAF algorithms has decreased to varying degrees. The RDAP-TSAP algorithm predicts, classifies, and aggregates user tasks to reduce the task resources required to push to the edge servers. During resource deployment, the delay threshold is judged to effectively reduce the occurrence of task delays, and then the Pareto optimizes task scheduling to make its TTDR always at a low level, with values ranging from 0.15% to 4.17%, which is an average reduction of 19.05% compared to the BAF algorithm, and an average reduction of 10.29% compared to the BAO algorithm. The TSAP algorithm lacks user task prediction and does not optimize task resource deployment of edge servers. However, it integrates the QoS and ESS for Pareto improvement during user task scheduling, and its TTDR value is relatively high, ranging from 11.22% to 22.71%, which is 4.28% lower than the BAF algorithm on average. The BAO algorithm jointly optimizes task caching and task offloading, and its TTDR value is 7.29% to 18.61%, which is an average reduction of 8.76% compared to the BAF algorithm. The TTDR value of the BAF algorithm is the highest, ranging from 15.31% to 28.51%, with an average value of 20.47%.

7. Conclusion

In this paper, we focused on resource deployment and task scheduling optimization using task prediction in edge-cloud collaborative computing for users and service providers. We presented an edge-cloud collaborative computing framework for edge-cloud IoT, and an approach for task prediction based on the two-dimensional time series. Then, for service-oriented tasks, we described our proposed RDAP. Using our approach, user tasks can be predicted based on the two-dimensional time series in the cloud service center, user tasks classified and aggregated, and the resources required for task operation pushed to the edge servers to improve average task hit rate and reduce server resource overhead. We also described our designed TSAP. In our approach, at the edge servers, the random greedy approximation algorithm is used to make Pareto improvement to both QoS and ESS, and the tangent point or intersection point of the two objective curves is used to optimize the task scheduling. Finally, the experimental evaluation shows that the RDAP-TSAP algorithm combining RDAP and TSAP realizes the comprehensive optimization of user's quality of service and effect of system service. Based on improving the average user task hit rate, the RDAP-TSAP algorithm has better ATCT, OSSE, and TTDR values than the TSAP, BAF, and BAO algorithms.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Key Program of the National Natural Science Foundation of China (61632009), Hunan Provincial Natural Science Foundation of China (2019JJ70057), Natural Science Foundation of Guangdong Province (2017A030308006), National Key Research and Development Program of China (2020YFB1005804), and Fundamental Research Funds for the Central Universities of Central South University (2018zzts180). The work of Kim-Kwang Raymond Choo is supported only by the Cloud Technology Endowed Professorship.

References

[1] Z. Cai, Z. He, X. Guan, and Y. Li, "Collective data-sanitization for preventing sensitive information inference attacks in social networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 4, pp. 577–590, 2018.

[2] X. Zhou, X. Yang, J. Ma, and K. Wang, "Energy efficient smart routing based on link correlation mining for wireless edge computing in IoT," *IEEE Internet of Things Journal*, 2021.

[3] Y. Xu, X. Yan, Y. Wu, Y. Hu, W. Liang, and J. Zhang, "Hierarchical bidirectional RNN for safety-enhanced B5G heterogeneous networks," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 4, pp. 2946–2957, 2021.

[4] J. Zhu, D. S. Chan, P. M. M. Suryanarayana, R. Natarajan, and H. Hu, "Improving web sites performance using edge servers in fog computing architecture," in *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, pp. 320–323, San Francisco, CA, USA, 2013.

[5] Y. Zhang and Y. Zhou, "Transparent computing: spatio-temporal extension on von neumann architecture for cloud services," *Journal of Tsinghua Science and Technology*, vol. 18, no. 1, pp. 10–21, 2013.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[7] T. Wang, Y. Mei, W. Jia, X. Zheng, G. Wang, and M. Xie, "Edge-based differential privacy computing for sensor-cloud systems," *Journal of Parallel and Distributed Computing*, vol. 136, pp. 75–85, 2020.

[8] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[9] Z. Cai, Z. Xiong, H. Xu, P. Wang, W. Li, and Y. Pan, "Generative adversarial networks," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–38, 2021.

[10] J. Zhang, Y. Wang, M. Long, J. Wang, and H. Wang, "Predictive recurrent networks for seasonal spatiotemporal data with applications to urban computing," *Chinese Journal of Computers*, vol. 43, no. 2, pp. 286–302, 2020.

[11] X. Zhu, Y. Luo, A. Liu, W. Tang, and M. Z. A. Bhuiyan, "A deep learning-based mobile crowdsensing scheme by predicting vehicle mobility," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4648–4659, 2021.

[12] Y. Sun, X. Yin, J. Jiang et al., "CS2P: improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of the 2016 ACM SIGCOMM Conference*, pp. 272–285, Florianopolis, Brazil, 2016.

[13] X. Zhou, X. Xu, W. Liang et al., "Intelligent small object detection for digital twin in smart manufacturing with industrial cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1377–1386, 2022.

[14] X. Zhu, Y. Luo, A. Liu, M. Z. A. Bhuiyan, and S. Zhang, "Multiagent deep reinforcement learning for vehicular computation Offloading in IoT," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9763–9773, 2021.

[15] T. Wang, Y. Lu, J. Wang, H. Dai, X. Zheng, and W. Jia, "EIHDP: edge-intelligent hierarchical dynamic pricing based on cloud-edge-client collaboration for IoT systems," *IEEE Transactions on Computers*, vol. 70, no. 8, pp. 1285–1298, 2021.

[16] X. Zhou, X. Xu, W. Liang, Z. Zeng, and Z. Yan, "Deep-learning-enhanced multitarget detection for end-edge-cloud surveillance in smart IoT," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12588–12596, 2021.

[17] X. Yan, Y. Xu, X. Xing, B. Cui, Z. Guo, and T. Guo, "Trustworthy network anomaly detection based on an adaptive learning rate and momentum in IIoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6182–6192, 2020.

[18] Z. Cai and X. Zheng, "A private and efficient mechanism for data uploading in smart cyber-physical systems," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 2, pp. 766–775, 2020.

[19] H. Mao, M. Schwarzkoopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *SIGCOMM '19: Proceedings of the ACM Special Interest Group on Data Communication*, pp. 270–288, Beijing, China, 2019.

[20] V. Jalaparti, P. Bodik, I. Menache, S. Rao, K. Makarychev, and M. Caesar, "Network-aware scheduling for data-parallel jobs: plan when you can," *ACM SIGCOMM Computer Communication Review*, vol. 45, pp. 407–420, 2015.

[21] X. Ren, G. Ananthanarayanan, A. Wierman, and M. Y. Hopper, "Decentralized speculation-aware cluster scheduling at scale," in *SIGCOMM '15: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 379–392, London, UK, 2015.

[22] A. Alashaikh, E. Alanazi, and A. Al-Fuqaha, "A survey on the use of preferences for virtual machine placement in cloud data centers," *ACM Computing Surveys*, vol. 54, no. 5, pp. 1–39, 2022.

[23] Y. Xu, J. Ren, Y. Zhang, C. Zhang, B. Shen, and Y. Zhang, "Blockchain empowered arbitrable data auditing scheme for network storage as a service," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 289–300, 2020.

[24] Z. Cai and Z. He, "Trading private range counting over big IoT data," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 144–153, Dallas, TX, USA, 2019.

[25] A. Chung, J. Park, and G. Robert, "Stratus: cost-aware container scheduling in the public cloud," in *SoCC '18: Proceedings of the ACM Symposium on Cloud Computing*, pp. 121–134, New York, NY, USA, 2018.

[26] X. Liu, J. Li, Z. Yang, and Z. Li, "A task collaborative execution policy in mobile cloud computing," *Chinese Journal of Computers*, vol. 40, no. 2, pp. 364–377, 2017.

[27] R. Nishtala, V. Petrucci, P. Carpenter, and M. Sjlander, "Twig: Multi-agent task management for colocated latency-critical cloud services," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 167–169, San Diego, CA, USA, 2020.

[28] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[29] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through vm migration and transmission power control," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810–819, 2017.

[30] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[31] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[32] J. He, D. Zhang, Y. Zhou, and Y. Zhang, "A truthful online mechanism for collaborative computation offloading in mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 7, pp. 4832–4841, 2020.

[33] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[34] T. Wang, M. Bhuiyan, G. Wang, L. Qi, J. Wu, and T. Hayajneh, "Preserving balance between privacy and data integrity in edge-assisted internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2679–2689, 2020.

[35] X. Zhou, W. Liang, S. Shimizu, J. Ma, and Q. Jin, "Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5790–5798, 2021.

[36] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: partial computation o_oading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016.

[37] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "O_oading in mobile edge computing: task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[38] J. Zhang, X. Hu, Z. Ning et al., "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2633–2645, 2018.

[39] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2018.

[40] Y. Ding, C. Liu, X. Zhou, Z. Liu, and Z. Tang, "A code-oriented partitioning computation offloading strategy for multiple users and multiple mobile edge computing servers," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4800–4810, 2020.

[41] Y. Xu, C. Zhang, G. Wang, Z. Qin, and Q. Zeng, "A blockchain-enabled deduplicatable data auditing mechanism for network storage services," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1421–1432, 2021.

[42] Z. Ge, J. Wang, C. Jiang et al., "Analysis of resource utilization of co-located clusters," *Chinese Journal of Computers*, vol. 43, no. 6, pp. 1103–1122, 2020.

[43] Aliyun, "Aliyun edge node service," (2021), http://www.aliyun.com/product/ens/.

[44] X. Zheng and Z. Cai, "Privacy-preserved data sharing towards multiple parties in industrial IoTs," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 968–979, 2020.

[45] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: evidence and implications," in *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, pp. 126–134, New York, NY, USA, 1999.

[46] X. Yan, Y. Xu, B. Cui, S. Zhang, T. Guo, and C. Li, "Learning URL embedding for malicious website detection," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6673–6681, 2020.

[47] Y. Xu, Q. Zeng, G. Wang, C. Zhang, J. Ren, and Y. Zhang, "An efficient privacy-enhanced attribute-based access control mechanism," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 5, pp. 1–10, 2020.

[48] X. Wang, Z. Wang, and F. Li, "Cache location selected algorithm for information-centric networking," *Journal of national university of defense technology*, vol. 41, no. 1, pp. 152–160, 2019.

[49] X. Zhou, W. Liang, K. Wang, R. Huang, and Q. Jin, "Academic influence aware and multidimensional network analysis for research collaboration navigation based on scholarly big data," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 1, pp. 246–257, 2021.

[50] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *IEEE INFOCOM 2018*, pp. 207–215, New York, NY, USA, 2018.

[51] M. Su, G. Wang, and R. Li, "Multidimensional qos cloud computing resource scheduling method based on stakeholder perspective," *The Journal of Communication*, vol. 40, no. 6, pp. 103–115, 2019.

[52] Y. Xu, C. Zhang, Q. Zeng, G. Wang, J. Ren, and Y. Zhang, "Blockchain-enabled accountability mechanism against information leakage in vertical industry services," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1202–1213, 2021.

[53] Q. Lu, X. Xu, L. Bass, L. Zhu, and W. Zhang, "A tail-tolerant cloud API wrapper," *IEEE Software*, vol. 32, no. 1, pp. 76–82, 2015.

[54] X. Zhou, Y. Li, and W. Liang, "CNN-RNN based intelligent recommendation for online medical pre-diagnosis support," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 3, pp. 912–921, 2021.

[55] C. Zhang, Y. Xu, Y. Hu, J. Wu, J. Ren, and Y. Zhang, "A blockchain-based multi-cloud storage data auditing scheme to locate faults," *Computing*, 2021.