Collaborative Resource Management and Workloads Scheduling in Cloud-Assisted Mobile Edge Computing across Timescales

Lujie Tang[1,2], Minxian Xu[1,*], Chengzhong Xu[3] and Kejiang Ye[1,*]

[1] *Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences*

[2] *University of Chinese Academy of Sciences*

[3] *University of Macau*

{*lj.tang, mx.xu, kj.ye*}*@siat.ac.cn, czxu@um.edu.mo*

***Abstract*—Due to the limited resource capacity of edge servers and the high purchase costs of edge resources, service providers are facing the new challenge of how to take full advantage of the constrained edge resources for Internet of Things (IoT) service hosting and task scheduling to maximize system performance. In this paper, we study the joint optimization problem on service placement, resource provisioning, and workloads scheduling under resource and budget constraints, which is formulated as a mixed integer non-linear programming problem. Given that the frequent service placement and resource provisioning will significantly increase system configuration costs and instability, we propose a two-timescale framework for resource management and workloads scheduling, named RMWS. RMWS consists of a Gibbs sampling algorithm and an alternating minimization algorithm to determine the service placement and resource provisioning on large timescales. And a sub-gradient descent method has been designed to solve the workload scheduling challenge on small timescales. We conduct comprehensive experiments under different parameter settings. The RMWS consistently ensures a minimum 10% performance enhancement compared to other algorithms, showcasing its superiority. Theoretical proofs are also provided accordingly.**

***Index Terms*—Mobile Edge Computing, Service placement, Resource provisioning, Workloads scheduling, Across Timescales**

I. Introduction

In recent decades, the rapid development of IoT devices and network communication technologies have significantly contributed to the proliferation of IoT application services [1]–[3]. The emergence of IoT application services, such as smart transportation, smart factory [4], smart city [5], and smart home, etc, have captured the attention and interest of the general public. Typically, IoT devices are constrained by physical dimensions and battery life, making it challenging to meet the computational resource and capability requirements of different application services [6]. Cloud computing is a key solution to the resource constraints of IoT devices. However, the wide area network connecting traditional cloud servers to IoT devices can introduce unpredictable communication delays and jitter, potentially causing adverse effects on latency-sensitive application services.

As an emerging promising computing paradigm, Mobile Edge Computing (MEC) has attracted growing attention from both industry and academia [7]–[9]. The distributed architecture of MEC enables the deployment of services and data closer to end-users, leading to significant reductions in response latency and data transmission costs. The appeal of these advantages has prompted more service providers to transition their operations to MEC platforms to meet the growing demand for real-time services. However, compared to the cloud with elastic resource capacity, the resources in edge nodes are limited. Only a small number of services or applications can be accommodated on edge servers.

A more effective approach to address resource scarcity is to jointly utilize the computation, storage and communication resources of the nearby edge servers and the remote cloud server, maximizing edge-edge and edge-cloud cooperation, achieving load balancing and minimizing execution latency. Taking the cloud-assisted edge computing system architecture depicted in Figure 1 as an example, each edge node has its own region coverage restrictions and resource capacity due to hardware constraints. IoT devices in different regions will send service requests to the nearest edge node. If a device is unable to request a specified service, the request will be routed to neighboring edge servers capable of handling the service request or to the cloud server for processing. In Figure 1, it can be observed that different regions have different types and numbers of workload requests. This also requires us to dynamically adjust service placement and resource provisioning based on actual workload conditions to ensure service performance. In this regard, service providers need to focus on the following critical issues: **service placement**, **resource provisioning** and **workloads scheduling**. In general, service placement requires dynamically optimizing the placement of services on edge servers to better utilize edge server resources [10]–[13]. Resource provisioning requires the flexibility to adjust resource provisioning for each service to optimize overall system performance [14]–[16]. Additionally,

*Corresponding authors.

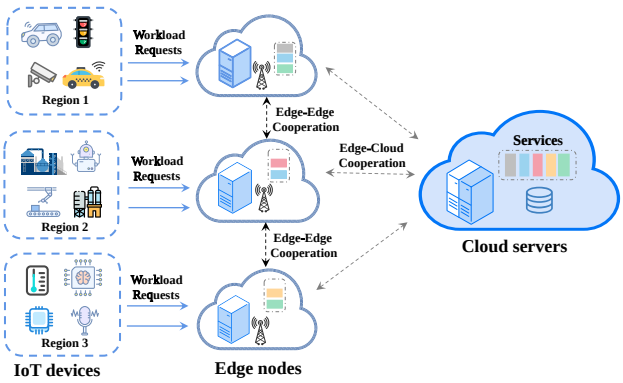


Fig. 1. System model.

to balance system workloads and improve system performance, workloads scheduling ensures that task requests are dynamically dispatched to appropriate edge nodes or remote cloud servers [17]–[19].

However, the aforementioned joint optimization problem may encounter some challenges. Firstly, the placement of services directly affects the resources they require and the scheduling of workloads, while resource provisioning should match the requirements of the services and their locations. This indicates a critical need to balance service placement, resource provisioning, and workloads scheduling. Furthermore, the popularity of services constantly fluctuates over time and space, and workload requests may exhibit significant short-term and long-term variations , which undoubtedly further increase the difficulty of joint optimization. The conventional approach is to over-provision resources to ensure service performance, but this can lead to high costs and low resource utilization. Therefore, it is essential for service operators to achieve efficient resource allocation for services. Finally, previous studies have frequently overlooked the essential requirement of managing system components across different timescales. In a real system, workload requests (e.g., HTTP) are easily transferred to cloud servers or edge nodes for processing due to their small data size. However, services placement may not be adjusted fast enough to meet the dynamic requirements of tasks. For example, configuring services on a new server refers to downloading the application container (including libraries and data stores) from the cloud server, adaptive configurations, initializing the service, etc, which requires a certain amount of time and additional costs. Therefore, it is natural to manage services and workload requests across two distinct timescales, i.e., service placement and resource provisioning can be handled on a larger timescale, whereas workload request scheduling can take place on a smaller timescale. This method not only reduces placement costs but also decreases the frequency of predicting future workloads. Multi-timescales approaches have demonstrated greater efficiency compared to single timescale methods in edge computing environments [20].

In this study, we tackle the issue above by devising a novel cloud-assisted mobile edge computing framework for dynamic resource management and workloads scheduling (RMWS). Firstly, we formulate the joint optimisation problem of service placement, resource provisioning and workloads scheduling to achieve minimised response time. Theoretically, the problem can be classified as a Mixed Integer Non-Linear Programming (MINLP) problem, which is difficult to solve directly. Then considering the complexity of the problem and the inherent differences in the optimisation periods of the three sub-problems [21], we have developed a two-timescale framework that aims to effectively address these issues by accounting for variations in their respective optimization frequencies. Besides, since service providers need to make a trade-off between cost and performance, our approach considers a scenario under budget constraints (including costs associated with storage and computation resources) that enable efficient allocation of computation and storage resources. Our contributions are summarized as follows:

- We investigate service placement, resource provisioning and workloads scheduling in a cloud-assisted mobile edge computing system. Then use queuing theory to characterize the response time of workloads and formulate the optimization goal as a mixed integer non-linear programming problem.
- We develop a joint optimisation framework under two-timescales to solve the MINLP problem. On large timescales, we employ a Gibbs sampling algorithm and an alternating minimisation method to determine the service placement and resource provisioning. On small timescales, we design a sub-gradient descent method to solve the workload scheduling challenge.
- We conduct comprehensive numerical simulations to evaluate the performance of the RMWS approach. The experiment results clearly demonstrate the superiority of our approach when compared with advanced methods.

II. Related Work

Recently, there has been a significant focus on enhancing the enforcement of service placement, resource allocation and computation offloading within edge-edge or edge-cloud collaboration environments. Based on the optimization with different timescales, we have categorized the relevant work into two buckets: single timescale and multiple timescales.

Joint Optimization on the Single Timescale. Edge-cloud collaboration is considered as an effective treatment to solve the problem of insufficient resources [22]–[24]. The authors in [22] presented a mechanism which utilizes parameterized deep Q network to jointly optimize service placement and computational resource allocation for task latency reduction. In [23], the authors focused on the limited resources in edge-cloud environments and proposed an optimisation method for joint communication and computational resource allocation.

There are also some studies focusing on Edge-Edge Collaboration [25], [26]. In [25], the authors explored the task offloading issue in ultra-dense networks, aiming to reduce delay while preserving the battery life of user equipment. The work [26] examined service placement from the perspective

of the service provider, aiming to minimize the expenses associated with service deployment in the hierarchical mobile edge computing network while fulfilling user requirements.

In reality, taking into account both horizontal cooperation among multiple edge nodes and vertical cooperation between edge nodes and cloud servers tends to result in more significant improvements in system performance. Therefore, researchers have actively explored various potentials in cloud-assisted mobile edge computing systems [27], [28]. The authors in [27] addressed the joint optimisation problem under uncertain demand in industrial cyber-physical system. They proposed an enhanced DQN algorithm for service placement and workload scheduling, and utilized convex optimization techniques to address resource allocation challenges. Poularakis et al [28] designed a service placement and request scheduling scheme with multidimensional (storage-computation-communication) constraints on the edge cloud.

However, in the above works, some issues are well-suited for optimization on a single timescale, such as task scheduling and task resource allocation. Nevertheless, regarding service deployment problems, we contend that it is not reasonable to approach service deployment and workloads scheduling optimization problems on the same timescale. On one hand, there is a significant difference in the amount of data between workloads and services, which results in varying levels of difficulty in their transmission and distribution. On the other hand, there is also a discrepancy in the initialization and startup times between them. If the deployment of services and resource allocation occur as frequently as the workloads scheduling, it can result in increased operational costs and heighten the instability of the system.

Joint Optimization across Multi-timescales. Multi-timescales optimal design has demonstrated greater efficacy in fulfilling practical requirements, with a scarcity of corresponding studies [20], [29], [30]. The work [20] proposed a two-timescale framework for joint service placement and request scheduling under resource constraints. Considering user deadline preference and edge clouds' strategic behaviors, Li et al. [29] designed a novel perspective on cost reduction by exploiting the spatial-temporal diversities in workload and resource cost among federated Edge Clouds. Wei et al. [30] studied the joint optimization problem of resource placement and task dispatching in edge clouds across multiple timescales under the dynamic status of edge servers.

In above studies, when considering the allocation of resources for service deployment, there is a common tendency to assume fixed CPU and storage configurations for the services. However, there is a lack of consideration from the perspective of the service provider in dynamically optimizing and adjusting the allocation of computation resources for services to meet the latency requirements of time sensitive tasks. In addition, a critical factor that has been overlooked is the budget constraints for service providers to purchase server resources for service deployment. Therefore, we shift our focus to a full exploration of joint optimisation problems across timescales in our research.

III. System Model and Problem Formulation

A. System Overview

We consider a typical cloud-assisted mobile edge computing system depicted in Figure 1, consisting of IoT devices, edge servers, and a remote cloud server. Define the set of edge servers as $\mathcal{L} = \{1, 2, ..., i, ..., L\}$ and the cloud server is indexed as $L + 1$. Each edge server i has a maximal computation capacity F_i and storage capacity M_i. The price of all storage and computation resources in edge server i are defined as P_i^m and P_i^f. The set of all possible services is $\mathcal{S} = \{1, 2, ..., s, ..., S\}$. For each service s, it needs an amount of storage resources m_s to cache related data which including software, databases and models, etc. We assume that c_s is the required CPU cycles to process corresponding tasks. Each service can be deployed in a specific set of edge servers and each edge server can also host various services depending on its available resources. In addition, all services will be deployed in the cloud. We use a Poisson process to characterize the dynamics of task arrivals at edge nodes with the arriving rate $n_{i,s}$ [17]. Then the total number of task requests for service s is noted as $n_s = \sum_{i=1}^{L} n_{i,s}$.

Service Placement Problem: Let binary variable $x_{i,s} \in \{0, 1\}$ indicator whether service s is placed at edge server i. The service placement decisions set is denoted as $\mathcal{X} = \{x_{i,s} | i \in \mathcal{L}, s \in \mathcal{S}\}$. Since the limited storage resource on edge node, the aservice placements at edge server i can not exceed the storage constrains: $\sum_{s=1}^{S} x_{i,s} m_s \leq M_i$.

Resource Provisioning Problem: Let $y_{i,s} \in [0, 1]$ represent the fraction of computation capacity allocated to service s. The resource allocation set is denoted as $\mathcal{Y} = \{y_{i,s} | i \in \mathcal{L}, s \in \mathcal{S}\}$. Here, $\sum_{s=1}^{S} y_{i,s} \leq 1$. Considering the budget constraints P_i^{bud} of edge server i , $p_i = \sum_{s=1}^{S} x_{i,s} \left(\frac{m_s}{M_i} p_i^m + y_{i,s} P_i^f \right) \leq P_i^{bud}$.

Workload Scheduling Problem: Let $z_{i,s} \in [0, 1]$ represent the workload ratio of the service s executed at edge service i. The workload scheduling policy is denoted as $\mathcal{Z} = \{z_{i,s} | i \in (\mathcal{L} \cup L + 1), s \in \mathcal{S}\}$. For each service s, the workload ratio needs to satisfied the following constrains: $\sum_{i=1}^{L+1} z_{i,s} = 1$.

B. Latency Model

The latency described in our model consists of both transmission latency and computation latency.

The Response Latency of Edge Servers: Since IoT devices are usually close to the wireless access point, we ignore the transmission latency between IoT devices and edge servers [31]. Assuming the transmission latency of the task served by service s between edge servers is ϕ_s, the transmission latency $T_{i,s}^{tran}$ can be obtained as:

$$T_{i,s}^{tran} = max\{z_{i,s} n_s - n_{i,s}, 0\} \cdot \phi_s. \tag{1}$$

We model the execution of tasks on the edge server as an M/M/1 queue and the computation latency $T_{i,s}^{com}$ of tasks served by service s which processed in edge server i can be computed as:

$$T_{i,s}^{com} = \frac{z_{i,s} n_s}{y_{i,s} F_i / c_s - z_{i,s} n_s / \Delta t}. \tag{2}$$

To ensure the stability of the queue or avoid infinite queue length, according to queue theory we have:

$$\frac{y_{i,s}F_i}{c_s} - \frac{z_{i,s}n_s}{\Delta t} > 0, \quad \forall i \in \mathcal{L}, \quad \forall s \in \mathcal{S}. \tag{3}$$

In brief, the total service response latency of the task offloading to the edge servers can be expressed as:

$$T_i = \sum_{s \in \theta_i} (T_{i,s}^{com} + T_{i,s}^{tran}), \tag{4}$$

where θ_i denotes the set of services placed on edge server i.

The Response Latency of Cloud Servers: Given a well-supplied cloud data center with sufficient computation resources and all required services, our focus is solely on the transmission latency for offloading tasks to the cloud server, disregarding computation latency [22], [32]. Let $\phi_{c,s}$ be the transmission delay for the task served by service s transferring to the cloud server. Therefore the total transmission latency of the task served by service s offloading to the cloud server can be obtained as:

$$T_{c,s}^{tran} = z_{L+1,s} n_s \phi_{c,s}. \tag{5}$$

The total response latency of the task offloading to the cloud server can be obtained as:

$$T_c = \sum_{s=1}^{S} T_{c,s}^{tran} = \sum_{s=1}^{s} z_{L+1,s} n_s \phi_{c,s}. \tag{6}$$

C. Problem Formulation

In our work, we jointly optimized the edge service placement, resource provisioning, and workloads scheduling policies, aiming at minimizing response time. Therefore, we formally define the problem as follows, denoted as **P1**. By observing, we identify it as a Mixed Integer Non-Linear Programming (MINLP) problem, which is challenging to solve directly.

$$\mathbf{P1}: \quad \min_{\mathcal{X},\mathcal{Y},\mathcal{Z}} T_{total} = \min_{\mathcal{X},\mathcal{Y},\mathcal{Z}} \sum_{i=1}^{L} T_i + T_c \tag{7}$$

$$s.t. \quad \sum_{s=1}^{S} x_{i,s} m_s \le M_i, \quad \forall i \in \mathcal{L} \tag{C1}$$

$$\sum_{s=1}^{S} y_{i,s} \le 1, \quad \forall i \in \mathcal{L} \tag{C2}$$

$$\sum_{i=1}^{L+1} z_{i,s} = 1, \quad \forall s \in \mathcal{S} \tag{C3}$$

$$P_i \le P_i^{bud}, \quad \forall i \in \mathcal{L} \tag{C4}$$

$$z_{i,s} n_s c_s - y_{i,s} F_i \Delta t < 0, \quad \forall i \in \mathcal{L}, \quad \forall s \in \mathcal{S} \tag{C5}$$

$$x_{i,s} \in \{0,1\}, \quad \forall i \in \mathcal{L}, \quad \forall s \in \mathcal{S} \tag{C6}$$

$$y_{i,s} \in [0,1], \quad \forall i \in \mathcal{L}, \quad \forall s \in \mathcal{S} \tag{C7}$$

$$z_{i,s} \in [0,1], \quad \forall i \in \mathcal{L}, \quad \forall s \in \mathcal{S} \tag{C8}$$

IV. Resource Management and Workloads Scheduling across Timescales

A. Problem Decoupling

Considering the complexity of the problem **P1**, we decoupled this problem and designed a two-timescale framework as displayed in Figure 2. The variables related to the time frame are denoted with the superscript f and subscript *frame* and those related to the time slot are denoted with the superscript t and subscript *slot*.

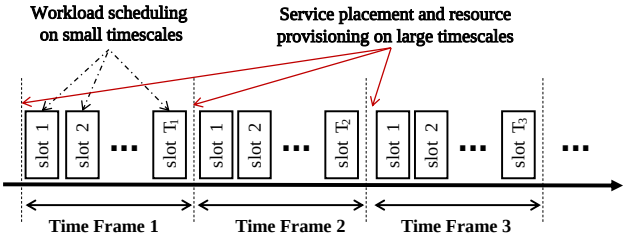


Fig. 2. Joint optimization across two-timescale.

At the start of time frame f, we solve problem **P1** in an iterative manner with the predicted request numbers $n_{i,s} = n_{i,s}^f$, $n_s = n_s^f$ and $\Delta t = \Delta t^f$. Note that only $\mathcal{X}_{frame}$ and $\mathcal{Y}_{frame}$ will be used in the time frame f, and $\mathcal{Z}_{frame}$ will not be used for actual scheduling (named **shadow workload scheduling**), but it plays an important role in assessing the given service placement and resource allocation strategies. At the start of time slot t, the request numbers $n_{i,s} = n_{i,s}^t$, $n_s = n_s^t$ and $\Delta t = \Delta t^t$, based on the optimal $\mathcal{X}_{frame}^*$ and $\mathcal{Y}_{frame}^*$ determined by time frame f, we perform optimal workload scheduling $\mathcal{Z}_{slot}^*$ in each time slot.

On large timescales, we decompose problem **P1** into several sub-problems and solve them using a two-layer iterative algorithm. In the outer layer, we update service placement decisions based on Gibbs sampling [33]. In the inner layer, we solve resource provisioning and shadow workload scheduling with alternating minimization algorithm. On small timescales, the service placement decisions and resource allocation are given and problem **P1** is reduced to the workload scheduling problem. we solve it based on the sub-gradient method. The flowchart of the RMWS algorithm is depicted in Figure 3.

B. service placement algorithm based on Gibbs sampling

Gibbs sampling is a Markovian Monte Carlo method commonly employed for dealing with multidimensional random variables. It can be introduced by the properties of Markov chains and transition probability matrix to conclude that its sampling distribution eventually converges to the joint distribution [34]. The core concept of Gibbs sampling is to randomly change one of the variable while keeping the rest of variables unchanged, repeating the iterations until convergence.

In the outer layer, we use the idea of Gibbs sampling to find the optimal service placement decisions with objective value of **P1**. We consider the process of updating the service placement decisions as a L-dimensional Markov chain in which the ith

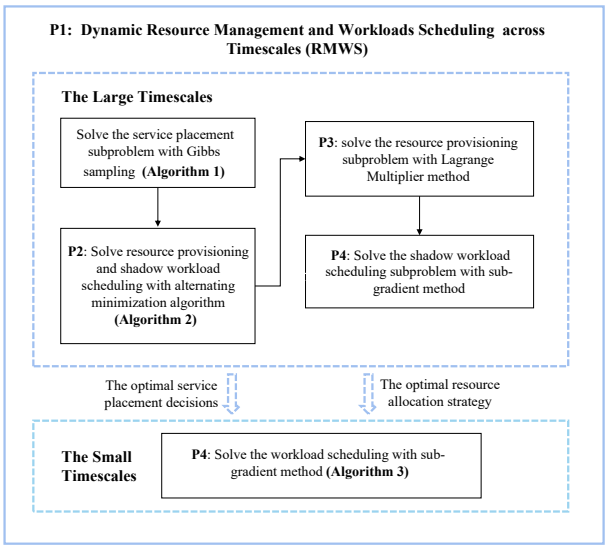


Fig. 3. The flowchart of the RMWS algorithm

dimension corresponds to ith edge server service placement decision and the algorithm is shown in **Algorithm 1**. In each iteration, the algorithm randomly selects an edge server k to update its service placement decision x_k to $x_k^{\#}$ while ensuring that the rest edge servers remain unchanged. Thus the optimization problem **P1** can be converted into the following optimization problem **P2** with the given service placement decisions of all edge servers:

$$\mathbf{P2}: \quad \min_{\mathcal{Y},\mathcal{Z}} T_{total} = \min_{\mathcal{Y},\mathcal{Z}} \sum_{i=1}^{L} T_i + T_c, \tag{8}$$
$$s.t. \quad C2 - C5, \quad C7 - C8$$

The problem **P2** will be addressed in the Section IV-C. And after solving it, we can obtain the optimal objective value ϑ (defined as $\vartheta = \min_{\mathcal{Y},\mathcal{Z}} T_{total}$). The ϑ will change to $\vartheta^{\#}$ when the edge server k updates decision x_k to $x_k^{\#}$ with the probability $\rho = \frac{1}{1+e^{(\vartheta^{\#}-\vartheta)/\omega}}$ and keep unchanged with probability $(1-\rho)$, where ω is a smooth parameter and $\omega > 0$.

Theorem 1. As the value of ω decreases, *Algorithm 1* is more likely to converge to the global optimal for problem **P1** and will converge to the global optimal solution with probability 1 when $\omega \to 0$.

Proof. Please see Appendix A.

C. Optimal resource provisioning and shadow workload scheduling with alternating minimization algorithm

Once the service placement decisions are determined in the outer layer, we face the task of resolving the resource allocation and workload scheduling problem, referred to as **P2** in the inner layer. It can be proved that the optimization problem **P2** is a non-convex problem with two variables $(y_{i,s}, z_{i,s})$ and solving it directly remains a challenging task. Thus, we propose an alternating minimization algorithm, whose detailed procedure is shown in **Algorithm 2**.

When we fix the variable $z_{i,s}$ as a constant value $z_{i,s}^*$, then the problem **P2** can be converted into the following problem

Algorithm 1 Service Placement Based on Gibbs Sampling

Input:

F_i, M_i, p_i^f, p_i^m, p_i^{bud}, m_s, c_s, $n_{i,s}$

Output:

The optimal service placement decisions $\mathcal{X}_{frame}^m$ and resource allocation strategy $\mathcal{Y}_{frame}^m$

Initialize service placement decisions $\mathcal{X}_{frame}^0$
for each iteration m = 1,2,... **do**
  Randomly select an edge server $k \in \mathcal{L}$ and an available service placement decision $x_k^{\#} \in \mathcal{X}_k$;
  if $x_k^{\#}$ is feasible **then**
    Use **Algorithm 2** to compute $\mathcal{Y}_{frame}^{m-1}$, $\mathcal{Z}_{frame}^{m-1}$ and the corresponding ϑ by solving **P2**, based on $(x_1^{m-1}, ..., x_k^{m-1}, ..., x_L^{m-1})$.
    Use **Algorithm 2** to compute $\mathcal{Y}_{frame}^{\#}$, $\mathcal{Z}_{frame}^{\#}$ and the corresponding $\vartheta^{\#}$ by solving **P2**, based on $(x_1^{m-1}, ..., x_k^{\#}, ..., x_L^{m-1})$.
    Let $\rho = \frac{1}{1+e^{(\vartheta^{\#}-\vartheta)/\omega}}$.
    $x_k^m = x_k^{\#}$, $\mathcal{Y}_{frame}^m = \mathcal{Y}_{frame}^{\#}$, $\mathcal{Z}_{frame}^m = \mathcal{Z}_{frame}^{\#}$ with the probability ρ.
    $x_k^m = x_k^{m-1}$, $\mathcal{Y}_{frame}^m = \mathcal{Y}_{frame}^{m-1}$, $\mathcal{Z}_{frame}^m = \mathcal{Z}_{frame}^{m-1}$ with the probability $1-\rho$.
  end if
  if the stopping criterion is satisfied **then**
    Return $\mathcal{X}_{frame}^m$, $\mathcal{Y}_{frame}^m$
  end if
end for

P3 with respect to $y_{i,s}$. it is easy to prove that this problem is a convex optimization problem, and classical Karush-Kuhn-Tucker (KKT) conditions [35] can be applied to find a closed-form solution. See *Sub-section D* for more details.

$$\mathbf{P3}: \quad \min_{\mathcal{Y}} T_{total} = \min_{\mathcal{Y}} \sum_{i=1}^{L} T_i + T_c, \tag{9}$$
$$s.t. \quad C2, \quad C4 - C5, \quad C7$$

Similarly, when we fix the variable $y_{i,s}$ as a constant value $y_{i,s}^*$, then the optimization problem **P2** can be converted into the following problem **P4** with respect to $z_{i,s}$. It is easy to prove that this problem is a convex optimization problem, and the sub-gradient descent method can be applied to find the optimal solution. See *Sub-section E* for more details.

$$\mathbf{P4}: \quad \min_{\mathcal{Z}} T_{total} = \min_{\mathcal{Z}} \sum_{i=1}^{L} T_i + T_c, \tag{10}$$
$$s.t. \quad C3, \quad C5, \quad C8$$

Theorem 2. Problem **P2** is a non-convex optimization problem for a given optimal service placement decisions $\mathcal{X}^*$.

Proof. Please see Appendix B.

Theorem 3. The **P3** and **P4** are the convex optimization problems.

Proof. Please see Appendix C.

Algorithm 2 Alternating Optimization-Based Algorithm

Input:

The service placement decisions $\mathcal{X}_{frame}$, task request numbers in time frame $n_{i,s}^f$, error tolerance threshold ϵ

Output:

The optimal resource allocation strategy $\mathcal{Y}_{frame}^n$ and shadow workload scheduling $\mathcal{Z}_{frame}^n$ under the service placement decisions $\mathcal{X}_{frame}$.

Initialize $\mathcal{Z}_{frame}^0$.
Obtain $\mathcal{Y}_{frame}^0$ based on the Lagrange multiplier method.
Calculate $f(\mathcal{Z}_{frame}^0)$ and $g^{(0)}$ according to Eq.(10) and Eq.(16)
for each iteration n = 1,2,... **do**
  Update the shadow workload scheduling vector by Eq. (15) and perform the weighting operation to satisfy constraint C3 and C8.
  Obtain resource allocation strategy $\mathcal{Y}_{frame}^n$ based on the Lagrange multiplier method.
  Calculate $f(\mathcal{Z}_{frame}^n)$ and $g^{(n)}$ according to Eq.(10) and Eq.(16)
  if $|f(\mathcal{Z}_{frame}^n) - f(\mathcal{Z}_{frame}^{n-1})| \leq \epsilon$ **then**
    Return $\mathcal{Y}_{frame}^n$, $\mathcal{Z}_{frame}^n$
  end if
end for

D. Optimal resource provisioning with Lagrange Multiplier method

According to **Theorem 3**, problem **P3** is can be solved by the Lagrange multiplier method [36]. The Lagrangian function for problem **P3** is established as follows:

$$L(f(\mathcal{Y}),\lambda,\mu) = f(\mathcal{X}^*,\mathcal{Z}^*,\mathcal{Y}) + \sum_{i=1}^{L}\lambda_i(\sum_{s\in\theta_i} y_{i,s} - 1) + \sum_{i=1}^{L}\mu_i\left[\sum_{s\in\theta_i}\left(\frac{m_s}{M_i}P_i^m + y_{i,s}P_i^f\right) - P_i^{bud}\right]. \tag{11}$$

where λ and μ are the Lagrangian multiplier. Then as shown in **Theorem 4**, we can derive the optimal solution by using KKT conditions.

Theorem 4. The optimal solution of problem **P3** is given as follows.

Case 1: if $\Gamma_i < 1$, the optimal solution is given by

$$y_{i,s}^* = \frac{\sqrt{z_{i,s}^* n_s c_s}(\Gamma_i F_i\Delta t - \sum_{s\in\theta_i} z_{i,s}^* n_s c_s)}{\sum_{s\in\theta_i}\sqrt{z_{i,s}^* n_s c_s}F_i\Delta t} + \frac{z_{i,s}^* n_s c_s}{F_i\Delta t}. \tag{12}$$

Case 2: if $\Gamma_i \geq 1$, the optimal solution is given by

$$y_{i,s}^* = \frac{\sqrt{z_{i,s}^* n_s c_s}(F_i\Delta t - \sum_{s\in\theta_i} z_{i,s}^* n_s c_s)}{\sum_{s\in\theta_i}\sqrt{z_{i,s}^* n_s c_s}F_i\Delta t} + \frac{z_{i,s}^* n_s c_s}{F_i\Delta t}, \tag{13}$$

where $\Gamma_i = \left(p_i^{bud} - \sum_{s\in\theta_i}\frac{m_s}{M_i}P_i^m\right)/P_i^f$.

Proof. Please see Appendix D.

E. Optimal shadow workload scheduling with sub-gradient method

The key challenge of Problem **P4** is that Eq. (10) is continuous but non-differential (or non-smooth) at $z_{i,s}n_s = n_{i,s}$. As a result, it is difficult to use the traditional KKT conditions directly and find a closed-form solution. For non-differential convex problems, a common approach is the sub-gradient descent method [37]. Therefore, we design an efficient algorithm based on sub-gradient descent. The sub-gradient of optimization objective **P4** is represented as : $\frac{\partial f(\mathcal{Z})}{\partial z_{i,s}}$

$$= \begin{cases} \frac{n_s c_s y_{i,s}^* F_i\Delta t^2}{(y_{i,s}^* F_i\Delta t - z_{i,s}n_s c_s)^2} + n_s\phi_s & z_{i,s}n_s > n_{i,s}, i\in\mathcal{L} \\ \left[\frac{n_s c_s y_{i,s}^* F_i\Delta t^2}{(y_{i,s}^* F_i\Delta t - z_{i,s}n_s c_s)^2}, \right. & \\ \quad \left. n_s\phi_s + \frac{n_s c_s y_{i,s}^* F_i\Delta t^2}{(y_{i,s}^* F_i\Delta t - z_{i,s}n_s c_s)^2}\right] & z_{i,s}n_s = n_{i,s}, i\in\mathcal{L} \\ \frac{n_s c_s y_{i,s}^* F_i\Delta t^2}{(y_{i,s}^* F_i\Delta t - z_{i,s}n_s c_s)^2} & z_{i,s}n_s < n_{i,s}, i\in\mathcal{L} \\ n_s\phi_{c,s} & i = L+1 \end{cases} \tag{14}$$

The problem **P4** can be solved by sub-gradient descent method in the following iteration:

$$z_{i,s}^{(n+1)} = z_{i,s}^{(n)} - \alpha_n g^{(n)}, \tag{15}$$

where $z_{i,s}^{(n)}$ denotes the value of the nth iteration of $z_{i,s}$, $\alpha_n > 0$ denotes the step size of the nth iteration and g is the sub-gradient of the function $f(\mathcal{Z})$ at $z_{i,s}^{(n)}$, it can be written as

$$g = \begin{cases} \partial f(\mathcal{Z}), & \text{subject to C5} \\ \partial(z_{i,s}n_s c_s - y_{i,s}^* F_i\Delta t), & \text{if } (z_{i,s}n_s c_s - y_{i,s}^* F_i\Delta t) > 0. \end{cases} \tag{16}$$

Notice that the constraints C3 and C8 will be satisfied by weighted operation, and $\partial(z_{i,s}n_s c_s - y_{i,s}^* F_i\Delta t)$ is utilized as the obstacle function.

F. Workloads Scheduling on Small Timescales

So far, we have discussed how to deploy service placement and allocate resources on a large timescale. The next matter that needs to be determined is the workload scheduling problem for each time slot. The optimal service placement decisions $\mathcal{X}_{frame}^*$ and resource allocation $\mathcal{Y}_{frame}^*$ will be given at the start of every time frame *f*. Then the problem **P1** will be simplified to the problem **P4** at the start of every time slot *t*. It can also be solved by sub-gradient method, which is summarized in **Algorithm 3**.

The complexity of RMWS: In **P3**, $y_{i,s}^*$ can be computed by the closed-form expressions, whose complexity can be neglected. In **P4**, the complexity of the sub-gradient descent algorithm mainly related to the number of iterations. Based on the proof in [37], it has a polynomial time complexity of $O(1/\epsilon^2)$. Therefore, the overall complexity of **Algorithm 3** and **Algorithm 2** is $O(1/\epsilon^2)$. The complexity of **Algorithm 1** is highly correlated with the number of Gibbs sampling iterations M and the complexity of **Algorithm 2**, which is $O(M/\epsilon^2)$.

Algorithm 3 Workloads scheduling on small timescales

Input:

$\mathcal{X}^*_{frame}$ and $\mathcal{Y}^*_{frame}$ in the time frame f, service request numbers $n^t_{i,s}$ in the time slot t.

Output:

The optimal workload scheduling $\mathcal{Z}^n_{slot}$ in the time slot t.

1: Set service placement $\mathcal{X}^*_{frame}$ and resource allocation strategy $\mathcal{Y}^*_{frame}$.
2: Initialize the workload scheduling $\mathcal{Z}^0_{slot}$.
3: Calculate $f(\mathcal{Z}^0_{slot})$ and $g^{(0)}$ according to Eq.(10) and Eq.(16)
4: **for** each iteration n = 1,2,... **do**
5: Update the workload scheduling vector by Eq. (15)
6: Calculate $f(\mathcal{Z}^n_{slot})$ and $g^{(n)}$ according to Eq.(10) and Eq.(16)
7: **if** $|f(\mathcal{Z}^n_{slot}) - f(\mathcal{Z}^{n-1}_{slot})| \leq \epsilon$ **then**
8: Return $\mathcal{Z}^n_{slot}$
9: **end if**
10: **end for**

V. Performance Evaluations

A. Experiment Setup

Parameter Setting: The default settings of parameters in our simulations are collected in Table I. We note that the values chosen for the parameters are practical and are widely used in existing work [14], [26], [38] and the price of edge server resource refers to Alibaba Cloud servers[1].

TABLE I
SIMULATION SETUP AND SYSTEM PARAMETERS

Parameters	Values
Number of edge servers, L	4
Number of services, S	10
Edge server storage resource, M_i	[50,200]GB
Edge server computation resource, F_i	[50,150] GHz
Service storage requirement, m_s	[10,40]GB
Service computation requirement, c_s	[0.1,0.5] GHz
Edge server storage resource price, P^m_i	[10, 40] CNY/hour
Edge server computation resource price, P^f_i	[10,50] CNY/hour
zipf distribution coefficient, e	0.6
budget coefficient, μ	0.7
Smooth parameter, ω	0.001

Service Request Demand: We assume that the service request demands of each edge server follow Zipf distribution, witch are consistent with the other researches [14]. The Zipf's law shows that the probability for a single request to the type s service is $p_s = \frac{1}{s^e H}$ for all services, i.e. p_s is the popularity of service s, where $H = \sum_{s=1}^{|S|} \frac{1}{s^e}$, $1 \leq s \leq |S|$ and these services are ranked in their popularity.

Task Arrival Pattern: Mobile edge computing has not been widely deployed in practice, thus we can not trace the mobile task arrivals at edge server precisely. In our work, we use the mathematical traffic pattern to simulate the time-varying property of mobile task arrivals at edge servers [17]. Assume that each time frame contains 30 time slots and per slot length is one minute. In each time frame, the number of task requests at each edge server obeys a normal distribution with a mean of 600 tasks/slot and a variance of 20 tasks/slot.

[1]https://www.aliyun.com/price

B. Benchmark Algorithms

We evaluate the performance of our algorithm RMWS with following state-of-the-art algorithms:

1) **Cloud Processing Only (CPO)**: No services are placed on the edge servers and all requested tasks are offloaded to the cloud for processing.
2) **Fixed Service Placement (FSP) Algorithm** [39]: Services placed on edge servers are fixed without service placement optimization.
3) **Non-cooperation Service Placement (NSP) Algorithm** [22]: Each edge server performs independent service placement, with the workload either processed on local edge server or offloading to the cloud server and the edge-edge cooperation is disabled.
4) **Popular Service Placement (PSP) Algorithm** [40]: Services are placed on edge servers according to popularity. Workload scheduling and computation resource provisioning are jointly optimized.
5) **Edge-Edge Cooperation Scheduling with Equal Resource Allocation (EERA)** [41]: which disables edge-cloud cooperation and resource allocation optimization.
6) **Edge-Cloud and Edge-Edge Cooperation Scheduling with Equal Resource Allocation (ECEERA)** [38]: which disables resource allocation optimization.

C. Simulation Results

1) **Comparison under Different Budget Constraints.** To evaluate the impact of allocated budge, we enhance the budget constraint for each edge server, raising it from 50% to 90% of the total cost of computation and storage resources across all edge servers. Meanwhile, the remaining parameter values are kept constant. It can be seen from Figure 4, the response latency of workloads shows a tendency to decrease with the elevation of the service provider's budget. The expanded budget enhances the availability of resources on edge servers, facilitating the deployment of services and processing of tasks. Notably, the response time of the service remains unaffected by the change in budget when employing the CPO approach, as the workloads are processed in the cloud.

2) **Comparison under Different Numbers of Service Types.** From the Figure 5, we can see that as the number of service types increases, there is a gradual rise in the average response time for each algorithm. This is primarily due to the limited resources of the edge servers, which cannot accommodate the growing number of services for placement. Tasks associated with services that cannot be placed on the edge servers are consequently offloaded to the cloud, thus prolonging the task processing delay caused by edge-cloud transmission. When the number of applications is 12, the

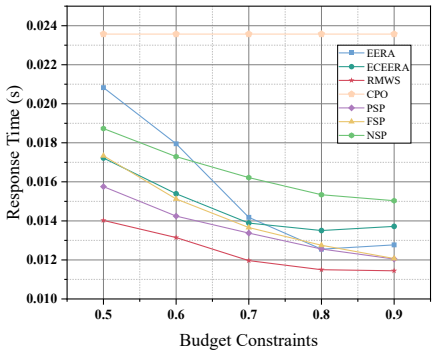


Fig. 4. Response time under different budget constraints

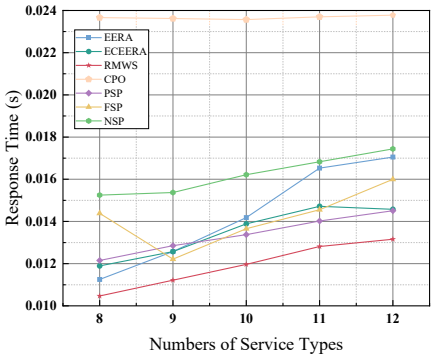


Fig. 5. Response time under different numbers of service types

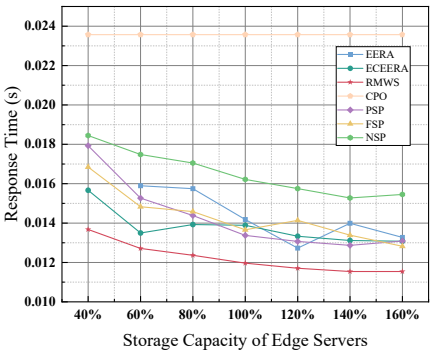


Fig. 6. Response time under different storage capacity of edge servers.

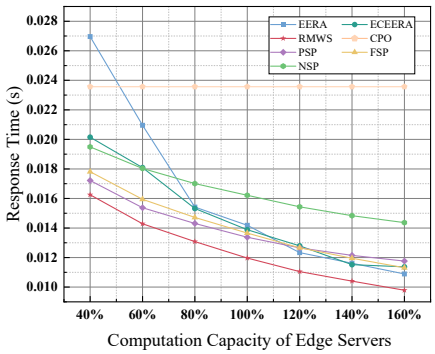


Fig. 7. Response time under different computation capacity of edge servers.

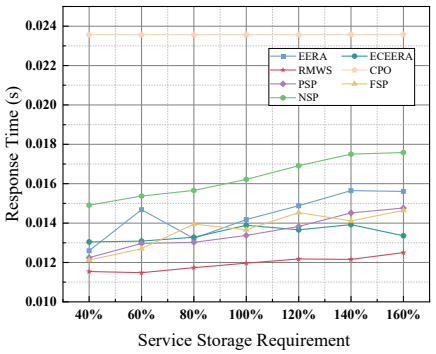


Fig. 8. Response time under different storage requirement of services.

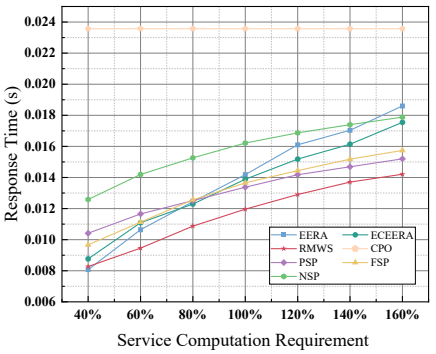


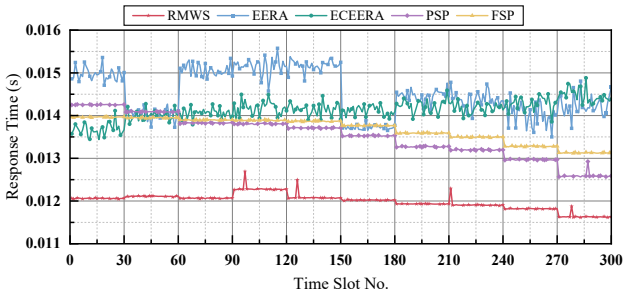
Fig. 9. Response time under different computation requirement of services.

RMWS reduces the response time by 10.7%-80.9% compared to other algorithms. The main reason is that RWMS can more effectively allocate resources on edge servers to various services and coordinate response latency between edge servers and cloud servers.

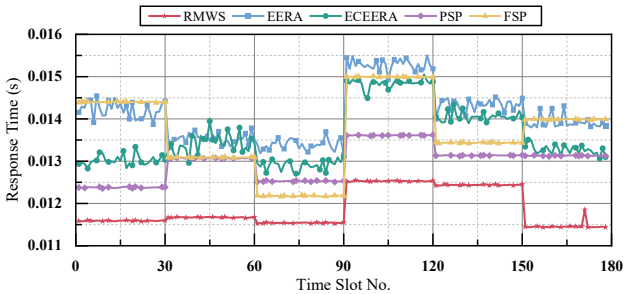
3) **Comparison under Different Storage and Computation Capacity of Edge Servers.** The capacity of computation and storage resources in edge servers dictates the number and distribution of services they can handle. In Figure 6 and Figure 7, It can be concluded that as the storage and computation capacity of edge servers increase, the service response time of each algorithm gradually decreases. Notably, the RMWS consistently maintains lower response delays by effectively balancing service popularity and resource availability within the edge-cloud cooperative system. Moreover, in the absence of coordination between edge and cloud, if edge servers have limited computation capabilities, processing the entire workload on edge servers can result in response delays surpassing those incurred by offloaded to cloud servers. Specifically, when the computation resources of the edge server are set to the default value of 40%, the average response latency relationship of each algorithm is EERA $>$ CPO $>$ NSP, FSP, PSP, ECEERA, RWMS. This indicates that in cases where edge resources are insufficient, the edge-cloud collaboration mechanism can effectively reduce task processing latency. When the computation and storage resources of the edge server are set to the default value of 160%, the performance of the non-collaborative service placement algorithm NSP is only better than CPO. This suggests that when edge resources are sufficient, cooperation among edge servers can also effectively reduce task processing time.

4) **Comparison under Different Storage and Computation Requirement of Services.** It can be seen from the Figure 8 and Figure 9, as the storage and computation demands of services increase, the number of services that edge servers can simultaneously deploy and tasks they can process both decrease. This will lead to an increase in service response time, but in such cases, RMWS consistently delivers optimal performance. The RMWS demonstrates notable efficiency by reducing service response latency by at least 10.6%, 11.9%, and 14.5% compared to other algorithms when services require 60%, 100%, and 140% of the default storage resources, respectively. Similarly, when services demand 60%, 100%, and 140% of the default computation resources, our algorithm achieves a reduction of at least 12.5%, 11.7%, and 7.2% in response latency compared to other algorithms, respectively. This indicates that the RMWS has more performance in resource allocation and utilization.

5) **Comparison under Different Service Popularity.** To validate the impact of service popularity on response time, we ensure that the total number of workload requests remains constant within each time frame, and adjust the popularity of



(a) The variations of Zipf distribution coefficients.



(b) The variations in the ranking of service popularity.

Fig. 10. Response time under different service popularity

services by modifying the coefficients of the Zipf distribution and the ranking of service popularity. In Figure 10(a), the experiment modifies the Zipf coefficient of service requests and randomly generates the Zipf distribution coefficients within 10 time frames: [0.23, 0.3, 0.41, 0.42, 0.46, 0.54, 0.64, 0.67, 0.76, 0.89]. In Figure 10(b), the experiment modifies the the ranking of service popularity and randomly generating 6 different rankings within 6 time frames. It can be observed that without resource configuration optimization (EERA and ECEERA), the response latency of tasks in each time slot fluctuates significantly. Other algorithms that have undergone resource configuration optimization can maintain a relatively stable state. Additionally, service response latency is somewhat affected by service popularity, but the RMWS algorithm overall remains in a better state.

In conclusion, the RMWS can achieve better performance based on dynamically changing workloads and resource requirements in terms of different metrics.

VI. Conclusions

This paper introduces a cloud-assisted edge computing framework focusing on service placement, resource management, and workloads scheduling. Considering the complexity of this problem and the distinct optimization cycles for different sub-problems, we propose a two-timescale optimization algorithm to minimize workload response time. Extensive simulations demonstrate the effectiveness and advantages of our algorithm in minimising the response delay. For future work, we will investigate more complex scenarios, considering the mobility of IoT devices in the workload scheduling process.

VII. Acknowledgments

This work is supported by National Key R & D Program of China (No. 2021YFB3300200), the National Natural Science Foundation of China (No. 62072451, 62102408, 92267105), Guangdong Basic and Applied Basic Research Foundation (No. 2024A1515010251, 2023B1515130002), Guangdong Special Support Plan (No. 2021TQ06X990), Shenzhen Basic Research Program (No. JCYJ20220818101610023), Shenzhen Industrial Application Projects of undertaking the National key R & D Program of China (No. CJGJZD20210408091600002).

Appendix

A. Proof of Theorem 1

Let $A = \{a_1, a_2, ..., a_M\}$ be the service placement decision space of all edge servers. At each iteration round, we randomly choose the edge server k and its service placement decision from A. Following the iterations of algorithm, the service placement decisions $\mathcal{X}$ evolves as a L-dimension Markov chain in which the ith dimension corresponds to the ith edge server service placement decision. For the convenience of proving, let $L = 2$, so it is a two-dimensional Markov chain and denoted as S_{x_1,x_2}. In each iteration, we change the service placement decision to $S_{x_1^*,x_2}$ with the following probability:

$$\begin{aligned}\Pr(S_{x_1^*,x_2}|S_{x_1,x_2}) &= \frac{1}{2M} \times \frac{1}{1+e^{(\vartheta(S_{x_1^*,x_2})-\vartheta(S_{x_1,x_2}))/\omega}} \\ &= \frac{1}{2M} \times \frac{e^{-\vartheta(S_{x_1^*,x_2})/\omega}}{e^{-\vartheta(S_{x_1^*,x_2})/\omega}+e^{-\vartheta(S_{x_1,x_2})/\omega}}.\end{aligned} \tag{17}$$

Let the stationary distribution be $\Pr^*$, according to the detailed balance condition,

$$\begin{aligned}&\Pr{}^*(S_{a_1,a_1})\Pr(S_{a_1,a_m}|S_{a_1,a_1}) \\ &= \Pr{}^*(S_{a_1,a_m})\Pr(S_{a_1,a_1}|S_{a_1,a_m}).\end{aligned} \tag{18}$$

it can be derived that:

$$\begin{aligned}&\Pr{}^*(S_{a_1,a_1}) \times \frac{1}{2M} \times \frac{e^{-\vartheta(S_{a_1,a_m})/\omega}}{e^{-\vartheta(S_{a_1,a_m})/\omega}+e^{-\vartheta(S_{a_1,a_1})/\omega}} \\ &= \Pr{}^*(S_{a_1,a_m}) \times \frac{1}{2M} \times \frac{e^{-\vartheta(S_{a_1,a_1})/\omega}}{e^{-\vartheta(S_{a_1,a_m})/\omega}+e^{-\vartheta(S_{a_1,a_1})/\omega}}.\end{aligned} \tag{19}$$

By observation the above equation, we can find Eq. (19) is symmetric and can be balanced if the stationary joint distribution $\Pr{}^*(\tilde{S}) = \gamma e^{-\vartheta(\tilde{S})/\omega}$ for arbitrary state $\tilde{S}$ in the strategy space Ω, where γ is a constant. Therefore, based on the probability conservation law, the stationary distribution can be expressed as

$$\Pr{}^*(S_{x_1,x_2}) = \frac{e^{-\vartheta(S_{x_1,x_2})/\omega}}{\sum\limits_{S_{\tilde{x}_1,\tilde{x}_2}\in\Omega} e^{-\vartheta(S_{\tilde{x}_1,\tilde{x}_2})/\omega}}. \tag{20}$$

Let $S_{x_1^*,x_2^*}$ be the globally optimal service placement decisions, thus $\vartheta(S_{x_1^*,x_2^*}) \leq \vartheta(S_{\tilde{x}_1,\tilde{x}_2})$ has always been established for any $S_{\tilde{x}_1,\tilde{x}_2} \in \Omega$. From Eq. (20), we observe

that $\Pr^*(S_{x_1^*,x_2^*})$ will increase with the decrease of ω and $\lim_{\omega\to 0}\Pr^*(S_{x_1^*,x_2^*})=1$, which proves that the gibbs sample algorithm will converges to the optimal state in probability.

The above analysis can be directly extended to the L-dimensional Markov chain.

B. Proof of Theorem 2

According to the convex optimization theory, If the Hessian is a positive definite matrix, the optimization objective function is a convex function [37]. By analysis, the optimization objective of problem **P2** can be transformed into:

$$f(\mathcal{X}^*,\mathcal{Y},\mathcal{Z})=\sum_{s=1}^{S}\left[(1-\sum_{i=1}^{L}z_{i,s})n_s\phi_{c,s}\right]+ \tag{21}$$
$$\sum_{i=1}^{L}\sum_{s\in\theta_i}\left[\frac{z_{i,s}n_s}{y_{i,s}F_i/c_s-z_{i,s}n_s/\Delta t}+max\{z_{i,s}n_s-n_{i,s},0\}\cdot\phi_s\right].$$

The Hessian of $f(y_{i,s},z_{i,s})$ is

$$\mathbf{H}=\begin{bmatrix}H_{11}&H_{12}\\H_{21}&H_{22}\end{bmatrix}=\begin{bmatrix}\dfrac{\partial^2 f(y_{i,s},z_{i,s})}{\partial^2 y_{i,s}}&\dfrac{\partial^2 f(y_{i,s},z_{i,s})}{\partial y_{i,s}\partial z_{i,s}}\\\dfrac{\partial^2 f(y_{i,s},z_{i,s})}{\partial z_{i,s}\partial y_{i,s}}&\dfrac{\partial^2 f(y_{i,s},z_{i,s})}{\partial^2 z_{i,s}}\end{bmatrix}, \tag{22}$$

$$\triangle_1=H_{11}=\frac{2z_{i,s}n_sc_sF_i^2\Delta t^3}{(y_{i,s}F_i\Delta t-z_{i,s}n_sc_s)^3}>0, \tag{23}$$

$$\triangle_2=\frac{-n_s^2c_s^2F_i^2\Delta t^4(y_{i,s}F_i\Delta t-z_{i,s}n_sc_s)^2}{(y_{i,s}F_i\Delta t-z_{i,s}n_sc_s)^6}<0. \tag{24}$$

where $\triangle_1=H_{11}$ and $\triangle_2=H_{11}H_{22}-H_{12}H_{21}$.

We can prove that the Hessian in Eq.(22) is indefinite by proving that the leading principal minors of **H** have both positive and negative,which are given by Eq. (23) and Eq. (24). Therefore, the problem **P2** is a non-convex optimization problem.

C. Proof of Theorem 3

The optimization objective of problem **P3** can be denoted as $f(\mathcal{X}^*,\mathcal{Z}^*,\mathcal{Y})$, then the second-order derivative of $f(\mathcal{X}^*,\mathcal{Z}^*,\mathcal{Y})$ is:

$$\frac{\partial^2 f(\mathcal{X}^*,\mathcal{Z}^*,\mathcal{Y})}{\partial y_{i,s}\partial y_{j,s}}=\begin{cases}\frac{2z_{i,s}^*n_sc_sF_i^2\Delta t^3}{(y_{i,s}F_i\Delta t-z_{i,s}^*n_sc_s)^3}\geq 0 & \text{if } i=j\\0 & \text{else}\end{cases}. \tag{25}$$

The parameters in Eq.(25) are all positive, so the Hessian matrix of the objective function $f(\mathcal{X}^*,\mathcal{Z}^*,\mathcal{Y})$ is positive definite. And problem **P3** can be defined as a convex optimization problem since its constraints are linear.

The optimization objective of problem **P4** can be denoted as $f(\mathcal{X}^*,\mathcal{Y}^*,\mathcal{Z})$, then the second-order derivative of $f(\mathcal{X}^*,\mathcal{Y}^*,\mathcal{Z})$ is:

$$\frac{\partial^2 f(\mathcal{X}^*,\mathcal{Y}^*,\mathcal{Z})}{\partial z_{i,s}\partial z_{j,s}}=\begin{cases}\frac{2y_{i,s}^*n_s^2c_s^2F_i\Delta t^2}{(y_{i,s}^*F_i\Delta t-z_{i,s}n_sc_s)^3}\geq 0 & \text{if } i=j\\0 & \text{else}\end{cases}. \tag{26}$$

The parameters in Eq.(26) are all positive, so the Hessian matrix of the objective function $f(\mathcal{X}^*,\mathcal{Y}^*,\mathcal{Z})$ is positive definite. And problem **P4** can be defined as a convex optimization problem since its constraints are linear.

D. Proof of Theorem 4

By using the KKT conditions, we can obtain that:

$$\begin{cases}\dfrac{\partial L(f(\mathcal{Y}),\lambda,\mu)}{\partial y_{i,s}}=0\\\lambda_i(\sum\limits_{s\in\theta_i}y_{i,s}-1)=0\\\mu_i\left[\sum\limits_{s\in\theta_i}\left(\dfrac{m_s}{M_i}P_i^m+y_{i,s}P_i^f\right)-P_i^{bud}\right]=0\\\lambda_i,\mu_i\geq 0\end{cases} \tag{27}$$

Based on the above formulas, we can derive that:

$$\begin{cases}y_{i,s}^*=\dfrac{\sqrt{\frac{z_{i,s}^*n_sc_sF_i\Delta t^2}{\lambda_i+\mu_iP_i^f}}+z_{i,s}^*n_sc_s}{F_i\Delta t}\\\sum\limits_{s\in\theta_i}y_{i,s}\leq\ 1\quad and\quad\sum\limits_{s\in\theta_i}y_{i,s}\leq\ \Gamma_i\end{cases} \tag{28}$$

where $\Gamma_i=\left(p_i^{bud}-\sum_{s\in\theta_i}\frac{m_s}{M_i}P_i^m\right)/P_i^f$.

Case 1: if $\Gamma_i<1$, it means that $\sum_{s\in\theta_i}y_{i,s}\leq\ \Gamma_i$ and $\lambda_i=0$.

$$\mu_i=\frac{(\sum_{s\in\theta_i}\sqrt{z_{i,s}^*n_sc_sF_i\Delta t^2})^2}{(\Gamma_iF_i\Delta t-\sum_{s\in\theta_i}z_{i,s}^*n_sc_s)^2P_i^f}, \tag{29}$$

$$y_{i,s}^*=\frac{\sqrt{z_{i,s}^*n_sc_s}(\Gamma_iF_i\Delta t-\sum_{s\in\theta_i}z_{i,s}^*n_sc_s)}{\sum_{s\in\theta_i}\sqrt{z_{i,s}^*n_sc_s}F_i\Delta t}+\frac{z_{i,s}^*n_sc_s}{F_i\Delta t}. \tag{30}$$

Case 2: if $\Gamma_i\geq 1$, it means that $\sum_{s\in\theta_i}y_{i,s}\leq\ 1$ and $\mu_i=0$.

$$\lambda_i=\left(\frac{\sum_{s\in\theta_i}\sqrt{z_{i,s}^*n_sc_sF_i\Delta t^2}}{F_i\Delta t-\sum_{s\in\theta_i}z_{i,s}^*n_sc_s}\right)^2, \tag{31}$$

$$y_{i,s}^*=\frac{\sqrt{z_{i,s}^*n_sc_s}(F_i\Delta t-\sum_{s\in\theta_i}z_{i,s}^*n_sc_s)}{\sum_{s\in\theta_i}\sqrt{z_{i,s}^*n_sc_sF_i\Delta t}}+\frac{z_{i,s}^*n_sc_s}{F_i\Delta t}. \tag{32}$$

This ends the proof.

REFERENCES

[1] S. Vimal, M. Khari, N. Dey, R. G. Crespo, and Y. H. Robinson, "Enhanced resource allocation in mobile edge computing using reinforcement learning based moaco algorithm for iiot," *Computer Communications*, vol. 151, pp. 355–364, 2020.

[2] S. C. Mukhopadhyay, S. K. S. Tyagi, N. K. Suryadevara, V. Piuri, F. Scotti, and S. Zeadally, "Artificial intelligence-based sensors for next generation iot applications: A review," *IEEE Sensors Journal*, vol. 21, no. 22, pp. 24 920–24 932, 2021.

[3] Y. Cui, F. Liu, X. Jing, and J. Mu, "Integrating sensing and communications for ubiquitous iot: Applications, trends, and challenges," *IEEE Network*, vol. 35, no. 5, pp. 158–167, 2021.

[4] F. Siqueira and J. G. Davis, "Service computing for industry 4.0: State of the art, challenges, and research opportunities," *ACM Computing Surveys (CSUR)*, vol. 54, no. 9, pp. 1–38, 2021.

[5] X. Xu, Q. Huang, X. Yin, M. Abbasi, M. R. Khosravi, and L. Qi, "Intelligent offloading for collaborative smart city services in edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 7919–7927, 2020.

[6] M. Goudarzi, M. Palaniswami, and R. Buyya, "Scheduling iot applications in edge and fog computing environments: a taxonomy and future directions," *ACM Computing Surveys*, vol. 55, no. 7, pp. 1–41, 2022.

[7] T. Wang, Y. Liang, X. Shen, X. Zheng, A. Mahmood, and Q. Z. Sheng, "Edge computing and sensor-cloud: Overview, solutions, and directions," *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–37, 2023.

[8] A. Brecko and et el., "Federated learning for edge computing: A survey," *Applied Sciences*, vol. 12, no. 18, p. 9124, 2022.

[9] J. Du, M. Xu, S. S. Gill, and H. Wu, "Computation energy efficiency maximization for intelligent reflective surface-aided wireless powered mobile edge computing," *IEEE Transactions on Sustainable Computing*, 2023.

[10] M. Goudarzi, H. Wu, M. Palaniswami, and R. Buyya, "An application placement technique for concurrent iot applications in edge and fog computing environments," *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1298–1311, 2020.

[11] T. Long, Y. Ma, Y. Xia, X. Xiao, Q. Peng, and J. Zhao, "A mobility-aware and fault-tolerant service offloading method in mobile edge computing," in *2022 IEEE International Conference on Web Services (ICWS)*. IEEE, 2022, pp. 67–72.

[12] K. Ray, "Adaptive service placement for multi-access edge computing: A formal methods approach," in *2023 IEEE International Conference on Web Services (ICWS)*. IEEE, 2023, pp. 14–20.

[13] M. Xu, Q. Zhou, H. Wu, W. Lin, K. Ye, and C. Xu, "Pdma: Probabilistic service migration approach for delay-aware and mobility-aware mobile edge computing," *Software: Practice and Experience*, vol. 52, no. 2, pp. 394–414, 2022.

[14] R. Li, Z. Zhou, X. Zhang, and X. Chen, "Joint application placement and request routing optimization for dynamic edge computing service management," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4581–4596, 2022.

[15] S. Smolka, L. Wißenberg, and Z. Á. Mann, "Edgedecap: An auction-based decentralized algorithm for optimizing application placement in edge computing," *Journal of Parallel and Distributed Computing*, vol. 175, pp. 22–36, 2023.

[16] J. Plachy, Z. Becvar, E. C. Strinati, and N. di Pietro, "Dynamic allocation of computing and communication resources in multi-access edge computing for mobile users," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 2089–2106, 2021.

[17] X. Ma, A. Zhou, S. Zhang, Q. Li, A. X. Liu, and S. Wang, "Dynamic task scheduling in cloud-assisted mobile edge computing," *IEEE Transactions on Mobile Computing*, 2021.

[18] H. Zhu, X. Li, L. Chen, and R. Ruiz, "Smart offloading computation-intensive & delay-intensive tasks of real-time workflows in mobile edge computing," in *2023 IEEE International Conference on Web Services (ICWS)*. IEEE, 2023, pp. 695–697.

[19] R. Zhang, R. Zhou, Y. Wang, H. Tan, and K. He, "Incentive mechanisms for online task offloading with privacy-preserving in uav-assisted mobile edge computing," *IEEE/ACM Transactions on Networking*, 2024.

[20] V. Farhadi, F. Mehmeti, T. He, T. F. La Porta, H. Khamfroush, S. Wang, K. S. Chan, and K. Poularakis, "Service placement and request scheduling for data-intensive applications in edge clouds," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 779–792, 2021.

[21] W. Fan, L. Zhao, X. Liu, Y. Su, S. Li, F. Wu, and Y. Liu, "Collaborative service placement, task scheduling, and resource allocation for task offloading with edge-cloud cooperation," *IEEE Transactions on Mobile Computing*, 2022.

[22] T. Liu, S. Ni, X. Li, Y. Zhu, L. Kong, and Y. Yang, "Deep reinforcement learning based approach for online service placement and computation resource allocation in edge computing," *IEEE Transactions on Mobile Computing*, 2022.

[23] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, 2019.

[24] W. Chu, P. Yu, Z. Yu, J. C. Lui, and Y. Lin, "Online optimal service selection, resource allocation and task offloading for multi-access edge computing: A utility-based approach," *IEEE Transactions on Mobile Computing*, 2022.

[25] 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.

[26] J. Huang, A. Zhou, and S. Wang, "Price-aware service deployment in hierarchical mobile edge computing," *IEEE Internet of Things Journal*, 2021.

[27] Y. Hao, M. Chen, H. Gharavi, Y. Zhang, and K. Hwang, "Deep reinforcement learning for edge service placement in softwarized industrial cyber-physical system," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5552–5561, 2020.

[28] 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 *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 10–18.

[29] Y. Li, W. Dai, X. Gan, H. Jin, L. Fu, H. Ma, and X. Wang, "Cooperative service placement and scheduling in edge clouds: A deadline-driven approach," *IEEE Transactions on Mobile Computing*, vol. 21, no. 10, pp. 3519–3535, 2021.

[30] X. Wei, A. M. Rahman, D. Cheng, and Y. Wang, "Joint optimization across timescales: Resource placement and task dispatching in edge clouds," *IEEE Transactions on Cloud Computing*, vol. 11, no. 1, pp. 730–744, 2021.

[31] 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 *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 10–18.

[32] X. Jie, T. Liu, H. Gao, C. Cao, P. Wang, and W. Tong, "A dqn-based approach for online service placement in mobile edge computing," in *Collaborative Computing: Networking, Applications and Worksharing: 16th EAI International Conference, CollaborateCom 2020, Shanghai, China, October 16–18, 2020, Proceedings, Part II 16*. Springer, 2021, pp. 169–183.

[33] A. Zhou, S. Li, and S. Wang, "Task offloading and resource allocation for container-enabled mobile edge computing," in *2021 IEEE International Conference on Services Computing (SCC)*. IEEE, 2021, pp. 222–232.

[34] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov chain Monte Carlo in practice*. CRC press, 1995.

[35] G. Gordon and R. Tibshirani, "Karush-kuhn-tucker conditions," *Optimization*, vol. 10, no. 725/36, p. 725, 2012.

[36] R. T. Rockafellar, "Lagrange multipliers and optimality," *SIAM review*, vol. 35, no. 2, pp. 183–238, 1993.

[37] S. Boyd and L. Vandenberghe, *Convex Optimization*. Convex Optimization, 2004.

[38] X. Ma, A. Zhou, S. Zhang, and S. Wang, "Cooperative service caching and workload scheduling in mobile edge computing," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2076–2085.

[39] A. Naouri, H. Wu, N. A. Nouri, S. Dhelim, and H. Ning, "A novel framework for mobile-edge computing by optimizing task offloading," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 13 065–13 076, 2021.

[40] C. Li, Q. Zhang, C. Huang, and Y. Luo, "Optimal service selection and placement based on popularity and server load in multi-access edge computing," *Journal of Network and Systems Management*, vol. 31, no. 1, p. 15, 2023.

[41] H. Ma, Z. Zhou, and X. Chen, "Leveraging the power of prediction: Predictive service placement for latency-sensitive mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6454–6468, 2020.