Online Service Placement, Task Scheduling, and Resource Allocation in Hierarchical Collaborative MEC Systems

An Du, Jie Jia, *Member, IEEE,* Schahram Dustdar, *Fellow, IEEE,* Jian Chen, *Member, IEEE,* and Xingwei Wang, *Member, IEEE*

***Abstract*—Mobile edge computing (MEC) pushes cloud computing capabilities to the network edge, which provides real-time processing and caching flexibility for service-based applications. Conventionally, the individual node solution is insufficient to tackle the increasing computation workload and provide diverse services, especially for unpredictable spatiotemporal service request patterns. To address this, we first propose a hierarchical collaborative computing (HCC) framework to serve users' demands by reaping sufficient computing capability in Cloud, ubiquitous service area in edge layer, and idle resources in device layer. To better unleash the benefits of HCC and pursue long-term performance, we investigate heterogeneity-aware resource management by collaborative service placement, task scheduling, and resource allocation both in-node and cross-node. We then propose an online optimization framework that first decouples the decisions across different slots. For each instant mixed integer non-linear programming problem, we introduce the surrogate Lagrangian relaxation method to reduce complexity and design hybrid numerical techniques to solve the subproblems. Theoretical analysis and extensive simulation results demonstrate the efficiency of the HCC framework in decreasing system cost on devices, and our proposed algorithms can effectively utilize the resources in the collaborative space to achieve the trade-off between system cost minimization and service placement cost stability.**

***Index Terms*—Heterogeneity-aware resources management, hierarchical collaborative computing, long-term performance, MEC, online framework.**

I. Introduction

In the past decade, wireless communications and networking have made tremendous advancements to adapt to the ever-growing number of mobile devices and mobile Internet traffic. The breakthroughs provide a high-rate and highly reliable air interface to run computing services ranging from real-time video streaming to surveillance at the remote cloud data centre [1]. However, the long propagation distance from the terminal devices (TDs) to the remote cloud centre remains a key bottleneck to fulfill the increasing expectations towards immersive quality of experience (QoE) for advanced applications, like augmented reality and autonomous driving. By transferring mobile computing, network control, and storage from the centralized cloud to the network edge, mobile edge computing (MEC) has become an integral component of the beyond fifth generation (5G) to offer low latency, context awareness, and mobility support [2]. Nonetheless, with a remarkable increase in data and heterogeneous service requests, any individual edge server (ES) with limited computation and storage capacity faces great challenges in satisfying the quality of service (QoS) requirements. Therefore, collaborative edge computing (CEC) has emerged as an attractive solution to enable scalable and resource-optimized operation, and support advanced applications in MEC systems [3].

The two state-of-the-art CEC frameworks have different architectures, which are suitable for different application scenarios. In horizontal collaboration, densely deployed base stations (BSs) support resource-limited ESs within specific geographic regions forming a federated resource pool to share placed services and collaboratively serve TDs' demands [4]. Benefiting from the expanded coverage and resources, TDs connected to any ES can instantly access the ubiquitous computing power and services, thereby enhancing service quality. Differently, vertical collaboration involves the layers with heterogeneous resource scales, such as the device layer, edge computing layer, and cloud layer [5]. Specifically, the cloud supports large-scale centralized computing, the edge provides agile data access and real-time computing, and the TDs achieve ubiquitous perception and local computing. Therefore, vertical collaboration facilitates meeting the demands of diverse applications regarding latency, service quality, operator profit, and other aspects by unified network resource management.

Practical MEC networks are typically characterized by large-scale deployment, variations in workload distribution over time and space, diverse application types, and different QoS requirements. These factors present significant challenges, including insufficient resource utilization, unsatisfactory service delays, and limited scalability in traditional CEC systems. Therefore, relying solely on collaborative computing, either horizontally or vertically, is inadequate. To this end, a hierarchical collaborative computing (HCC) solution becomes essential to pursue the enlarged and multi-level computing power, by reaping the specific benefits in different layers. Nonetheless, the widely distributed heterogeneous resources, in terms of locations, memory, and computing capabilities, increase the complexity of network management. Moreover,

A. Du, J. Jia (Corresponding author), J. Chen, X. Wang are with the School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China (e-mail: 2110663@stu.neu.edu.cn; jiajie@mail.neu.edu.cn; chenjian@mail.neu.edu.cn; wangxw@mail.neu.edu.cn). Schahram Dustdar is with the Distributed Systems Group, TU Wien, Vienna, 1040, Austria (e-mail: dustdar@dsg.tuwien.ac.at).

the rapid service response for complicated applications or real-time inference tasks, such as industrial robotics, voice assistants, and ad-targeting, requires the related libraries and well-trained models to be deployed in advance. However, constrained resources of individual ES naturally raise the problem of service placement, i.e., where to place each service and to reap the benefits of resources in the collaboration space. In addition, service placement determines which type of task to process, thus affecting task scheduling and resource allocation. Therefore, optimally placing services, scheduling tasks, and managing network resources are intractable but significantly crucial for improving overall system performance.

The problem of optimally placing services has been studied in several works such as [6], [7], yet customized service provisioning in the HCC framework still faces great challenges. One is the spatial coupling in the network caused by diverse services and heterogeneous network infrastructure. For each service request, the service demand and communication/computation resources become highly coupled. For example, self-driving requires more computing resources to achieve rapid response, while augmented reality requires a larger throughput for immersive experience. For each computing node, resource sharing leads to the coupling among multiple TDs. Therefore, HCC requires collaborative resource management, both in-node and cross-node, thus leading to a complex interaction setting and making the optimization problem thorny. The other challenge mainly lies in the temporal coupling, in which the time-based interests, e.g., rush hour traffic routing services, and channel conditions are time-varying and non-stationary, thus making the most of strategies that operate in static CEC scenarios impractical [8]–[11].

Motivated by the above considerations, we investigate the following problems: how to manage heterogeneous distributed resources uniformly? How can we efficiently distribute diverse service requests to the collaborative space without future information while maintaining asymptotical optimality? The problems are challenging and have not been well investigated hitherto. In this paper, we propose an HCC-assisted MEC system, which encompasses horizontal collaboration in both the device and edge layers for expanded service areas, and vertical end-edge-cloud collaboration for providing multi-level computing power. The systematic framework is illustrated in Fig. 1, in which the heterogeneous resource abstraction and hierarchical resource management are invoked for efficient resource utilization and QoS-guaranteed service provisioning.

Considering the diverse service demands are usually unknown/non-stationary and change spatially and temporally, we formulate the problem as a multi-stage stochastic optimization problem to minimize the long-term sum system cost of all TDs, subject to the time-averaged service placement cost of BSs, task delay requirements and hybrid resource capacity. The Lyapunov optimization and the Markov decision process (MDP) based approaches were proposed for dynamic resource allocation in the stochastic network [12]. However, the MDP-based method may exhibit the curse of dimensionality when the system enlarges. Aiming to minimize the system cost of TDs and ensure the long-term service placement cost budget of BSs, the Lyapunov optimization theory is more suitable since it

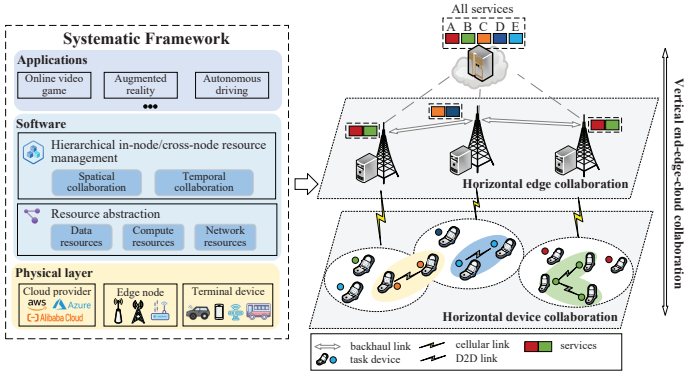


Fig. 1. The systematic framework of hierarchical collaborative MEC system.

can effectively incorporate long-term constraints into real-time optimizations, and make online decisions without requiring any a priori future information. Based on this, we develop an online optimization framework, OJSTR, to transform the time-averaged optimization problem into a queue stability problem. Then, a decoupled framework and hybrid-method-based iterative algorithms are proposed to jointly obtain service placement, task scheduling, and resource allocation decisions. The main contributions of this paper are summarized as follows.

- To provide satisfactory service for stochastical service requests cost-efficiently, we investigate a long-term joint service placement, task scheduling, uplink transmission power, and computational resources optimization problem in HCC systems, under the predefined long-term service placement cost budget on BSs and resource constraints in multiple dimensions.
- We propose an online optimization framework, which first applies the Lyapunov optimization technique for time decoupling and performance-cost trade-off. Then, to solve per-slot offline problems in low computational effort, we develop a two-loop optimization framework based on the surrogate Lagrangian relaxation method, where the outer loop obtains the proper Lagrangian multipliers and the inner loop solves two separable subproblems.
- We invoke dynamic programming algorithm for the service placement subproblem and a bilevel optimization framework for hierarchical task scheduling and resource allocation. We design a heuristic matching game for upper-level task scheduling. We devise a bi-section search algorithm and convex optimization method for lower-level resource allocation in edge collaboration and device collaboration, respectively.
- Numerical results demonstrate that the OJSTR can unleash the potential of the HCC system, and provide high-quality services cost-efficiently as well as guarantee the long-term budget. Moreover, it outperforms the other three benchmark schemes regarding different matrices.

The remainder of the paper is organized as follows. In Section II, we present a brief overview of the related literature. In Section III, we give the system model and formulate a long-term system cost minimization problem. In Section IV

and Section V, we develop an online optimization framework and its theoretical analysis. In Section VI, we provide extensive simulations and contrastive analysis. Finally, Section VII concludes this paper.

II. Related Work

With a remarkable increase in data and heterogeneous service requests generated by massive devices, collaborative edge computing and efficient resource management schemes have been widely investigated in recent years to alleviate the dilemma of computing restrictions of the MEC system with a single resource-constrained MEC server.

A group of existing works paid plenty of attention to vertical collaboration. For example, the authors in [13] studied the service caching problem in cloud-assisted MEC networks, in which the cloud is constituted as a powerful resort for a single resource-limited edge server. However, the architecture exhibits high communication overhead during the task offloading process between the TDs and the ES or cloud. To tackle this issue, the idle resources within the close vicinity TDs can be harvested by leveraging the Device-to-Device (D2D) communication technique [14]. Benefiting from this, collaboration among TDs was considered in [15], in which the computation tasks can be offloaded to the nearby TDs and the ES. In MEC systems, the authors in [8], [9], [16]–[18] investigated the D2D-assisted CEC, while limited to vertical end-edge collaboration or end-edge-cloud collaboration, engaging only single/independent ES. Therefore, apart from extending the MEC's capacity to other computing layers, densely deployed BSs can cooperate horizontally to enhance the service supply capability of the edge layer.

In the CEC framework, resource management schemes have been widely studied to ensure QoS demands and save system costs. For the D2D-assisted MEC system, the authors in [16], [17] studied the joint optimization of task scheduling(or partitioning) and the computing resources allocation to reduce the system delay. In addition, the authors in [18] further considered the bandwidth resources in the wireless access network. Differently, the authors in [8], [9] focused on the trade-off between performance and cost consumption by joint task offloading and heterogeneous resource allocation. Furthermore, the authors in [19] considered the service placement optimization problem in edge-cloud cooperation networks to minimize the task processing delay, while ignoring the service placement cost and system energy consumption. In addition, several works have focused on D2D-assisted heterogeneous collaborative edge caching [10], [20] and cache-enabled MEC networks [21] that combine D2D communication and edge collaboration. However, different from these works that pay more attention to storage resources, service placement requires various resource types for storing application services and performing atomic functionalities, thus making the above studies not applicable to the diverse service provisioning scenarios. Moreover, although several works designed efficient task offloading and service placement schemes in horizontal collaborative MEC systems [22], cloud-assisted MEC networks [6], [23], and D2D-assisted MEC networks [24], the efficient service provisioning in HCC-assisted MEC systems is typically challenging and still has not been well addressed.

Considering the time-varying and non-stationary environment, some of the existing works investigated dynamic resource management in CEC. For example, the authors in [25] investigated the computation offloading schemes in dynamic D2D-assisted communication architectures, and proposed an attention communication deep reinforcement learning algorithm to deal with a partially observable environment. The authors in [19] proposed a collaborative service placement, task scheduling, computing resource, and transmission rate allocation for the scenario with edge-cloud and edge-edge cooperation to minimize the total task processing delay while guaranteeing long-term task queuing stability. Nevertheless, collaborative service placement, task scheduling, and resource management in hierarchical collaborative MEC systems are still challenging and have not been well investigated hitherto.

III. System model

In this section, we describe the service placement, task scheduling, delay and cost model in the HCC system. The main symbols are summarized in Table I.

As depicted in Fig. 1, we consider a network $\mathcal{G} = (\mathcal{M} \cup \mathcal{N} \cup \{c\}, \mathcal{E})$, where $\mathcal{M} = \{1, 2, \ldots, M\}$ denotes the set of geographically distributed BSs equipped with ESs, $\mathcal{N} = \{1, 2, \ldots, N\}$ denotes the set of TDs, c is the remote cloud, and $\mathcal{E} = \{e_{i,j}, i, j \in \mathcal{S}, i \neq j\}$ is the set of links between BSs. Each ES $m \in \mathcal{M}$ is characterized by the tuple of resources in multiple dimension $\{W_m, F_m\}$, where W_m and F_m denote the maximum storage capacity and CPU computing capacity, respectively. Assume the system involves diverse services, exemplified by online gaming and voice assistants, that are indexed by $\mathcal{K} = \{1, 2, \ldots, K\}$. To simplify the system model, we assume the tasks in a specific service type are similar in request parameters [6], [7]. Each service $k \in \mathcal{K}$ is characterized by the tuple $\{w_k, d_k, c_k, \tau_k\}$, where w_k is the occupied storage capacity (bits), d_k is the data size of the service request (bits), c_k is the required number of CPU cycles, and τ_k is the delay upper bound.

Without loss of generality, each TD runs a wide spectrum of applications in a stochastic manner since TDs may gain or lose interest in some services. To better describe the time-varying nature, we assume the system runs within a finite time horizon, which is discretized into equal length time slots $\mathcal{T} = \{1, 2, \ldots, T\}$. Each TD $n \in \mathcal{N}$ is characterized by the tuple $\left\{P_n, F_n, \{K_n(t)\}_{t=1}^{T}\right\}$, where P_n is the maximum transmission power, F_n is the total CPU computing capacity, and $K_n(t) \in \mathcal{K}$ is the requested service type at time slot t. Moreover, we assume the geographically closed TDs are randomly moving within the coverage of different BSs. The location of TD and BS are denoted as l_n and l_m, respectively, and if $||l_n - l_m|| < R_m, m \in \mathcal{M}$, we suppose TD n is within the communication coverage of BS m, where R_m is the covering radius of m. Similarly, if $||l_n - l_i|| < R_i, i \in \mathcal{N}$, we suppose TD n is within the communication coverage of TD i, where R_i is the D2D communication range of the TD i. To ease representation, we denote Ω_m and Ω_n as the set of TDs covered by BS m and TD n, respectively.

TABLE I
IMPORTANT NOTATIONS IN OUR MODEL

Symbol	Definition
$\mathcal{M}$	The set of BSs integrating ESs, $m \in \mathcal{M}$.
$\mathcal{N}$	The set of TDs, $n \in \mathcal{N}$.
$\mathcal{K}$	The set of services, $k \in \mathcal{K}$.
$\mathcal{E}$	The set of wired links between BSs, $e_{i,j} \in \mathcal{E}$, $i, j \in \mathcal{S}$.
$w_{i,j}$	The bandwidth of outgoing link $e_{i,j}$.
W_m, F_m	Maximum storage and computing capacity of BS m.
P_n, F_n	Maximum transmission power and computing capacity of n.
K_n	Requested service type of TD n.
w_k	The occupied storage capacity of service k.
d_k	Data size of the service request of service k.
c_k	Required CPU cycles number of service k.
τ_k	Delay upper bound of service k.
$\widetilde{C}$	Time averaged budget of service placement budget.
C_m	Service placement cost for the ES m.
$\alpha_{k,m}$	Indicator of the service k is placed on m or not.
$\beta_{n,u}$	Indicator of the task of TD n is scheduled to node u or not.
p_n	The allocated transmission power of TD n.
$f_{n,u}$	The allocated computing resource of u for TD n.
$D_{n,u}$	Total delay for processing task of TD n on node u.
$E_{n,o}$	The transmission energy consumption for TD n.
$E_{n,c}$	The computing energy consumption for TD n.

A. Services placement

Since the device layer can only support small-scale computing and the TD may experience significant energy consumption in serving multiple applications, device collaboration requires the helper TD to possess the same service requests. Therefore, we mainly focus on the service placement at the edge layer.

To specify the placement of all services, we define a binary variable $\alpha_{k,m}(t)$ as the placement indicator. Specifically, $\alpha_{k,m}(t) = 1$ indicates that the service $k \in \mathcal{K}$ hosts on BS $m \in \mathcal{M}$ at time slot t. Although services can be dynamically placed among all BSs to deliver a higher QoS, the switching cost consumed by fetching services from the nearby BSs or cloud centre and cache updates cannot be ignored. To this end, we denote the service placement cost on ES m as

$$C_m(t) = \sum_{k=1}^{K} \left(1 - \alpha_{k,m}(t-1)\right) \alpha_{k,m}(t) c_{m,k}, \tag{1}$$

where $c_{m,k}$ is the cost of service k placing on BS m. In addition, $\left(1 - \alpha_{k,m}(t-1)\right) \alpha_{k,m}(t) = 1$ indicates that the service replacement occurs on BS m at time slot t. Considering the service provider generally operates within a long-term cost budget [26], we introduce the time-averaged service placement cost budget $\widetilde{C}$ over the whole period of T time slots as

$$C1: \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{m=1}^{M} C_m(t) < \widetilde{C}. \tag{2}$$

B. Hierarchical task scheduling

HCC supports arrival service requests to be executed across different layers. Therefore, whether and where to offload diverse tasks should be determined jointly by taking into account the service availability, achieved performance, and the required cost. Let $\mathcal{U} = \{\mathcal{M} \cup \mathcal{N} \cup c\}$ denote the set of computing nodes in collaboration space with the size of $|\mathcal{U}| = M+N+1$. Define a binary variable $\beta_{n,u}(t), n \in \mathcal{N}, u \in \mathcal{U}$ as the task scheduling indicator, and $\beta_{n,u}(t) = 1$ indicates that the service request of TD n is executed on the computing node u. In this paper, we assume each task is atomic and cannot be separated into subtasks. Therefore, each task can only be scheduled to one of the nodes in $\mathcal{U}$ for task processing, thus we have

$$C2: \begin{cases} \beta_{n,u}(t) \in \{0,1\}, \forall n \in \mathcal{N}, u \in \mathcal{U}, \\ \sum_{u\in\mathcal{U}} \beta_{n,u} = 1, \forall n \in \mathcal{N}. \end{cases} \tag{3}$$

In particular, $\beta_{n,n}(t) = 1$ represents that the service request of TD n will be processed locally.

Moreover, scheduling the service request of TD n to ES m requires the corresponding service has been placed on m, i.e., $\alpha_{K_n(t),m}(t) = 1$. Therefore, it also should be ensured that

$$C3: \beta_{n,m}(t) \leq \alpha_{K_n(t),m}(t), m \in \mathcal{M}. \tag{4}$$

C. Delay model

In device collaboration plane, each TD has its own service requests and may still have the potential to help others. Each TD adopts the Frequency Division Duplexing (FDD) technique to support transmitting and receiving tasks simultaneously [9]. To reduce communication interference, we adopt orthogonal frequency division multiple-access (OFDMA) and overlay D2D communication techniques for uplink transmission.

1) Device collaboration: We assume D2D transmission is completed in "one hop," which means the helper TD will not continue to offload after receiving the task. If $\beta_{n,i} = 1, i \in \mathcal{N} \setminus n$, the transmission delay, experienced by TD n to offloading packets to the helper TD i via uplink, can be expressed as

$$D_{n,i,o}^{D2D} = \frac{d_{K_n}(t)}{r_{n,i}(t)}, \tag{5}$$

where

$$r_{n,i}(t) = B_{n,i} \log_2 \left(1 + \frac{p_n h_{n,i}(t)}{\sigma^2}\right), \tag{6}$$

is the achieved uplink transmission rate. Specifically, $B_{n,i}$ is the allocated bandwidth, p_n is the transmission power of TD n, $h_{n,i}(t)$ is the channel gain between TD n and its helper, and σ^2 is the noise power. Noticeably, if the TD executes the task locally, we have $D_{n,n,o}^{D2D} = 0$.

2) Edge collaboration: Service requests scheduled to the edge layer can be processed on the associated BS in "one hop" or a non-local BS along the routing path [27]. Therefore, if $\beta_{n,i} = 1, i \in \mathcal{M}$, the transmission delay experienced by TD n is constituted by the uplink transmission delay and the backhaul delay. Specifically, the experienced uplink transmission delay for TD n is given by

$$D_{n,i,o}^{MEC} = \frac{d_{K_n}(t)}{r_{n,m_n}(t)}, \tag{7}$$

where r_{n,m_n} is the communication rate between TD n and its connected BS m_n. The consumed backhaul delay of TD n from the connected BS m_n to the computing node i can be represented as

$$D_{n,i,b}^{MEC} = \sum_{e_{j,k} \in \mathcal{P}_{m_n,i}} \frac{d_{K_n(t)}}{w_{j,k}}, \tag{8}$$

where $w_{j,k}, j, k \in \mathcal{S}$ is the bandwidth of outgoing link $e_{j,k} \in \mathcal{E}$ and $\mathcal{P}_{m_n,i}$ is the shortest routing path from the connected BS m_n to target BS i.

3) Total delay: Considering the computing resources can be shared for processing the offloaded tasks in parallel, the processing delay for TD n on node u can be calculated as

$$D_{n,u,c} = \frac{c_{K_n(t)}}{f_{n,u}(t)}, \tag{9}$$

where $f_{n,u}(t)$ is the allocated computing resource for TD n.

According to the aforementioned definitions, the total delay under different task schedule decisions is given by

$$D_{n,u} = \begin{cases} D^{D2D}_{n,u,o} + D_{n,u,c}, & u \in \mathcal{N}, \\ D^{MEC}_{n,u,o} + D^{MEC}_{n,u,b} + D_{n,u,c}, & u \in \mathcal{M}. \end{cases} \tag{10}$$

D. Cost Model

Considering the energy-limited TDs and instances tenanting from the cloud, we aim to reduce the system cost on devices for the service provisioning in HCC systems, which mainly includes energy consumption cost and cloud tenancy cost.

1) Device energy consumption: In the HCC system, the energy consumption on TDs consists of transmission energy consumption for offloading service requests and computing energy consumption for providing computing services. Specifically, for TD n, they can be calculated as

$$E_{n,o} = p_n \left(\sum_{m \in \mathcal{M}} \beta_{n,m} D^{MEC}_{n,m,o} + \sum_{i \in \mathcal{N}} \beta_{n,i} D^{D2D}_{n,i,o} \right), \tag{11}$$

and

$$E_{n,c} = \sum_{i \in \mathcal{N}} \beta_{i,n}(t) \kappa f^2_{i,n} c_{K_n}(t), \tag{12}$$

respectively. $\kappa > 0$ is the energy efficiency parameter determined by the structure of devices [6].

2) Cloud tenancy cost: If no BSs or TDs can provide satisfactory services, cloud computing is viewed as a last resort, while accessing services unavoidably causes a high tenancy cost. In this paper, we adopt on-demand instances as the representative of cloud service provision that charged proportionally to the usage of computation resources without upfront fees or long-term commitments [28].

3) Total cost: Based on the above definition, the overall cost for each TD n can be calculated as

$$\Psi_n(t) = \psi_n E_n(t) + \psi_c \beta_{n,c}(t) \tilde{f}_n(t), \tag{13}$$

where ψ_n is the price for unit energy of TD n, ψ_c is the price of on-demand instances, $E_n = E_{n,o} + E_{n,c}$ is the total energy consumption on TD n, and $\tilde{f}_n(t) = \frac{c_{K_n}(t)}{\tau_{K_n}(t)}$ is the tenanted computing resources from cloud to ensure the lowest requirements of service quality.

E. problem formulation

Our aim is to minimize the total cost of the TDs and provide satisfactory services over a long time span by jointly optimizing the service placement, task scheduling, uplink transmission power, and computing resource allocation. We must also ensure the resource capacity in multiple dimensions and the long-term service placement cost budget. We thus formulate the problem as

$$\textbf{(P0)}: \min_{\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{p},\mathbf{f}} \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{n=1}^{N} \Psi_n(t) \tag{14}$$

$$\text{s.t.} \quad C1 - C3, \tag{14a}$$

$$\sum_{k=1}^{K} \alpha_{k,m}(t) w_k \le W_m, m \in \mathcal{M}, \tag{14b}$$

$$\alpha_{k,m}(t) = \{0,1\}, \forall k, m, \tag{14c}$$

$$\sum_{u=1}^{U} \beta_{n,u}(t) D_{n,u} \le \tau_{K_n(t)}, u \in \mathcal{M} \cup \mathcal{N} \tag{14d}$$

$$\sum_{n=1}^{N} \beta_{n,u}(t) f_{n,u}(t) \le F_u, u \in \mathcal{M} \cup \mathcal{N}, \tag{14e}$$

$$0 \le p_n \le P_n, \forall n, \tag{14f}$$

where constraint (14b) guarantees the hosted services on each BS cannot exceed its storage capacity; constraint (14d) ensures the task deadline for each TD; constraint (14e) ensures the computing resources allocated for all served TDs by the node u must not exceed its computation capacity F_u; constraint (14f) is the transmission power constraint for all TDs.

IV. LYAPUNOV OPTIMIZATION MODEL TRANSFORMATION

To make our model more realistic and expandable, we assume there is no prior statistical information about the system dynamics. Moreover, optimizing the time-averaged sum cost minimization problem with the long-term placement cost constraint is difficult. To address these issues, we adopt Lyapunov optimization technology to convert problem **(P0)** into a series of per-slot queue stability control problems. Firstly, we define a virtual queue $Q(t)$ to measure the exceeded cost of service placement by the end of time slot t. Its updated equation can be represented as

$$Q(t+1) = \max\left\{ Q(t) - \widetilde{C} + C(t), 0 \right\}, \tag{15}$$

where $C(t) = \sum_{m=1}^{M} C_m(t)$ is the total service placement cost at time slot t. We assume the initial queue length is 0, i.e., $Q(0) = 0$. It can be implied that the larger value of $Q(t)$ means the consumed placement cost has far exceeded the long-term budget. To ensure the inequality (2) holds, we have:

Lemma 1. *The constraint $C1$ will always be satisfied when the virtual queue is mean rate stable, i.e., $\lim_{T\to\infty} \frac{1}{T}\mathbb{E}\{Q(T)\} = 0$.*

Proof. Please see the detailed proof in the Appendix A. □

To make the virtual queue stable, we first invoke the quadratic Lyapunov function and define it as

$$L(Q(t)) = \frac{1}{2} Q(t)^2. \tag{16}$$

Then, we define the conditional Lyapunov drift over one slot as

$$\Delta(Q(t)) = \mathbb{E}\{ L(Q(t+1)) - L(Q(t)) \,|Q(t)\}, \tag{17}$$

which is the expected change of the Lyapunov function from one slot to the next with the given queue state. Intuitively, minimizing $\Delta(Q(t))$ would push the virtual cost queue to a lower congestion region. Correspondingly, the original problem has been decomposed into a series of per-slot optimization problems. Aiming to coordinate the system cost consumption while guaranteeing queue stability, we use a drift-plus-penalty function according to Lyapunov optimization theory [29]:

$$\Delta\left(Q(t)\right) + V\sum_{n=1}^{N}\Psi_n, \tag{18}$$

where $V \geq 0$ is a control parameter to adjust the attention on the service placement cost queue backlogs and the system cost consumption. Moreover, the function (18) can provide the following performance guarantee:

Theorem 1. *For arbitrary queue backlogs $Q(t)$ over all possible control strategies, the drift-plus-penalty function has the following upper bound:*

$$B + \sum_{n=1}^{N} V\mathbb{E}\left\{\Psi_n|Q(t)\right\} + Q(t)\mathbb{E}\left\{C(t) - \widetilde{C}|Q(t)\right\}, \tag{19}$$

where $B = \frac{1}{2}\left(C_{max}^2 + \widetilde{C}^2\right)$ is a finite constant.

Proof. The detailed proof is shown in Appendix B. □

With **Theorem 1**, minimizing the drift-plus-penalty function is equivalent to minimizing the right side of (19). Hence, after removing the constant that is not concerned with the control decision, the optimal solution can be obtained at each time slot by optimizing the following problem

$$\textbf{(P1)}: \min_{\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{p},\mathbf{f}} Q(t)C(t) + \sum_{n=1}^{N} V\Psi_n(t) \tag{20}$$

$$\text{s.t.} \quad C2, C3, (14b) - (14f). \tag{20a}$$

V. Joint Optimization Algorithm Design

The per-slot deterministic problem **(P1)** is a mixed integer non-linear programming (MINLP) problem, which jointly determines the binary decisions of service placement and task scheduling, and the system resource allocation decisions. The commonly used integer programming algorithms, such as the cutting plane and branch and bound methods, typically require prohibitively high computational complexity. To address this challenge, we propose an efficient solution framework based on hybrid optimization methods, as depicted in Fig. 2. First, we introduce a surrogate Lagrangian relaxation method to reduce computational requirements and develop a two-loop optimization framework, as discussed in Section V-A. The inner loop optimizes the Lagrangian relaxation problem for given Lagrangian multipliers, and the outer loop performs multipliers update via a novel surrogate subgradient method. Moreover, the service placement subproblem and the joint task scheduling and resource allocation subproblem are solved via dynamic programming in Section V-B and a bilevel optimization algorithm in Section V-C, respectively. By solving the subproblems in two loops iteratively until it converges, the near-optimal solution can thus be obtained.

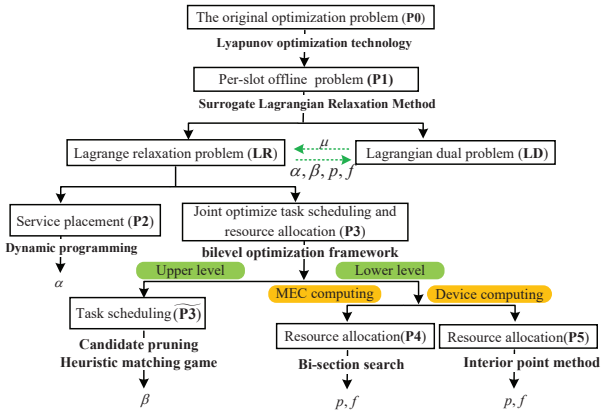


Fig. 2. The schematics of the proposed OJSTR.

A. Surrogate Lagrangian Relaxation Method

For problem **(P1)**, service placement $\boldsymbol{\alpha}$ is coupled with the task scheduling $\boldsymbol{\beta}$ in constraint $C3$. Therefore, Lagrangian relaxation with separability support can be leveraged to obtain a near-optimal solution in a computationally efficient manner. To simplify the representation in $C3$, we invoke a binary constant $\theta_n^k = \{0, 1\}$ to indicate that if the TD n requests service k, thus we have $\alpha_{K_n(t),m} = \sum_{k=1}^{K}\theta_n^k\alpha_{k,m}$. To dualize and penalize $C3$ into the objective function with Lagrangian multiplier $\boldsymbol{\mu} = \{\mu_{n,m}\}$, the Lagrangian function becomes

$$L(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{p},\mathbf{f},\boldsymbol{\mu}) = Q(t)C(t) + \sum_{n=1}^{N} V\Psi_n + \sum_{m=1}^{M}\sum_{n=1}^{N}\mu_{n,m}\left(\beta_{n,m} - \sum_{k=1}^{K}\theta_n^k\alpha_{k,m}\right). \tag{21}$$

Then, the Lagrange relaxation of **(P1)** is given by

$$\textbf{(LR)}: z(\boldsymbol{\mu}) = \min_{\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{p},\mathbf{f}} L(\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{p},\mathbf{f},\boldsymbol{\mu}) \tag{22}$$

$$\text{s.t.} \quad C2, (14b) - (14f). \tag{22a}$$

Obviously, the relaxed **(LR)** can be rewritten as two individual subproblems, i.e., one is the service placement subproblem, and the other is the joint optimization of task scheduling and resource allocation subproblem. Given the multipliers, these subproblems are easier than **(P1)**, and can be solved in a computationally efficient manner by our proposed methods in Sections V-B and V-C. The resolution of the original problem is thus performed through a two-level iterative approach, where the low level consists of solving individual subproblems and the high level performs Lagrange multipliers update.

Moreover, the Lagrangian dual problem is described as

$$\textbf{(LD)}: \max_{\boldsymbol{\mu}\geq 0} z(\boldsymbol{\mu}). \tag{23}$$

Since the dual function is the pointwise infimum of a family of affine functions of $\boldsymbol{\mu}$, it is always concave regardless of the characteristics of the original problem [30]. Additionally, since the original problem is mixed-integer, the dual function is non-smooth [31]. Therefore, the problem **(LD)** is convex and non-smooth. To solve it, the subgradient method is the most widely

used, which requires the relaxed subproblems to be fully optimized. This is time-consuming, especially as the problem size increases. Correspondingly, the surrogate subgradient method is developed, which only needs an approximate optimization for one subproblem to obtain a proper surrogate subgradient [32]. However, its convergence proof requires the optimal dual value, which is generally impractical. Inspired by [31], we adopt the surrogate Lagrangian relaxation method to guarantee convergence without requiring the optimal dual value, which selects stepsizes in a way that distances between Lagrange multipliers at consecutive iterations decrease. Specifically, the multipliers are updated by

$$\mu_{m,n}^{i+1} = \left[\mu_{m,n}^{i} + s^i \tilde{g}_{m,n}(\sum_{k=1}^{K} \theta_n^i \alpha_{k,m}^i, \beta_{m,n}^i)\right]^+, \forall s. \quad (24)$$

Here, i is the iterate number and $\tilde{g}_{m,n}$ is the surrogate subgradient direction by performing an approximate optimization of **(LR)**. s^i is the stepsize, which is updated by

$$s^i = \delta_i \frac{s^{i-1}\|\tilde{g}_{m,n}(\alpha_{m,n}^{i-1}, \beta_{m,n}^{i-1})\|}{\|\tilde{g}_{m,n}(\alpha_{m,n}^{i}, \beta_{m,n}^{i})\|}, 0 < \delta_i < 1. \quad (25)$$

To ensure that the multipliers converge to optimal dual value $\boldsymbol{\mu}^*$, the stepsize parameter δ_i can be updated as [31]

$$\delta_i = 1 - \frac{1}{Mi^p}, p = 1 - \frac{1}{i^r}, \quad (26)$$

where $M \geq 1$ and $0 \leq r \leq 1$. With the novel stepsizing formula, we adopt the approximate optimization of **(LR)** in an interleaved manner, where one subproblem is solved at a time to update multipliers. The main procedure is summarized in **Algorithm 1**, in which $Flag$ is used for marking the subproblem that is required to be optimized in each iteration to obtain a surrogate subgradient.

Proposition 1. *With the stepsizing formula (25) and the δ_k update formula (26), the Lagrange multipliers in guaranteed to converge within limited iterations.*

Proof. The detailed proof is shown in Appendix C. □

Algorithm 1 Per-slot optimization algorithm

Input: The current virtual queue state Q and service placement scheme $\boldsymbol{\alpha}(t-1)$.

Output: The near-optimal service placement $\boldsymbol{\alpha}$, task scheduling $\boldsymbol{\beta}$, and resource allocation $\mathbf{p}$, $\mathbf{f}$.

Initialization: Lagrangian multipliers $\boldsymbol{\mu}^0$, step size $\boldsymbol{s}^0$, subproblem record $Flag = True$.
By optimizing problem **(LR)** with $\boldsymbol{\mu}^0$, obtain $\{\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0\}$ and surrogate subgradient $\tilde{\boldsymbol{g}}(\boldsymbol{\alpha}^0, \boldsymbol{\beta}^0)$.
Iterate number $i = 1$.
repeat
  Update stepsize-setting parameters $\boldsymbol{\delta}^i$ by using (26).
  Given $\boldsymbol{\delta}^i, \boldsymbol{\alpha}^i, \boldsymbol{\beta}^i$, update step size $\boldsymbol{s}^i$ by using (25).
  For the given $\boldsymbol{s}^i, \boldsymbol{\alpha}^i, \boldsymbol{\beta}^i$, update lagrangian multiplier $\boldsymbol{\mu}^{i+1}$ by using (24).
  if $Flag = True$ **then**
    Given multipliers $\boldsymbol{\mu}^{i+1}$, solve the service placement subproblem **(P2)** by the dynamic programming method and obtain $\boldsymbol{\alpha}^{i+1}$
    Update $Flag = False$.
  else
    Given multipliers $\boldsymbol{\mu}^{i+1}$, solve the joint optimization subproblem of task scheduling and resource allocation **(P3)** by **Algorithm 2**, and obtain $\boldsymbol{\beta}^{i+1}, \mathbf{p}^{i+1}, \mathbf{f}^{i+1}$.
    Update $Flag = True$.
  end if
  Calculate the objective function value $Obj(i)$.
  $i = i + 1$.
until $i > I$ or $|Obj(i) - Obj(i-1)| \leq T_{th1}$

B. Service Placement Subproblem

With the Lagrangian multipliers, the service placement subproblem can be rearranged into the following concise form

$$\textbf{(P2)}: \min_{\boldsymbol{\alpha}} \sum_{m=1}^{M}\sum_{k=1}^{K} \sigma_{m,k}\alpha_{k,m} - \sum_{m=1}^{M}\sum_{n=1}^{N}\mu_{n,m}\sum_{k=1}^{K}\theta_n^k\alpha_{k,m} \quad (27)$$

$$\text{s.t.} \quad (14b), (14c), \quad (27a)$$

where $\sigma_{m,k} = Q(t)\left(1 - \alpha_{k,m}(t-1)\right)c_{m,k}$ is a constant that does not affect the placement decision-making. Moreover, the problem (**P2**) can be further separated into M individual subproblems. For each BS m, its optimization objective function can be reformulated as

$$\max_{\boldsymbol{\alpha_m}} \sum_{k=1}^{K}\left(\sum_{n=1}^{N}\mu_{n,m}\theta_n^k - \sigma_{m,k}\right)\alpha_{k,m} \quad (28)$$

Intuitively, the problem can be reduced from a 0-1 knapsack problem, corresponding to placing K services on the BS m at a maximized total profit, subject to the total storage constraint W_m, where the placement cost of service k is represented as $\sum_{n=1}^{N}\mu_{n,m}\theta_n^k - \sigma_{m,k}$. To solve the problem effectively, we adopt the dynamic programming method, and the detail process is omitted for page limits. By leveraging primal decomposition and the dynamic programming method, the computational complexity of solving service placement sub-problems for all BSs is $\mathcal{O}_s = \mathcal{O}(K\sum_{m\in\mathcal{M}} W_m)$.

C. Joint Task Scheduling and Resource Allocation Subproblem

From (**LR**), the other relaxed part is the joint task scheduling and resource allocation optimization subproblem, which can be formulated as

$$\textbf{(P3)}: \min_{\boldsymbol{\beta},\mathbf{p},\mathbf{f}} \sum_{n=1}^{N} V\psi_n\left(E_{n,o} + E_{n,c}\right) + V\psi_c\beta_{n,c}(t)\tilde{f}_n(t) + \sum_{n=1}^{N}\sum_{m=1}^{M}\mu_{n,m}\beta_{n,m} \quad (29)$$

$$\text{s.t.} \quad C2, (14d) - (14f), \quad (29a)$$

in which the control strategy includes both binary-valued $\boldsymbol{\beta}$ and the continuous-valued $\mathbf{p}, \mathbf{f}$. By analyzing **(P3)**, we observe that the resource allocation problem can be efficiently

solved once the task scheduling is determined. Meanwhile, the performance of schedule decisions can be evaluated accurately only when the optimal resource allocation scheme is obtained. Hence, the problem can be transformed into a bilevel optimization problem [33], in which the task scheduling is regarded as the upper optimization problem to minimize total cost, and the resource allocation optimization problem is optimized at the lower level to minimize the total energy cost. By fully considering the dependency, we propose a matching game-based bilevel optimization algorithm, which involves a matching theory-based heuristic method for the upper level, and a hybrid resource allocation algorithm for the lower level. The details are presented in **Algorithm 2**.

1) Heuristic matching-based task scheduling: Resource allocation decisions made in lower-level are only related to $\beta_{n,u}, u \in \mathcal{N} \cup \mathcal{M}$. With resource allocation $\mathbf{p}, \mathbf{f}$ and Lagrangian multiplier $\boldsymbol{\mu}$, the objective function of the optimization problem in the upper level can thus be transformed into

$$\begin{aligned}\widetilde{\mathbf{(P3)}}: \min_{\boldsymbol{\beta}} & \sum_{n=1}^{N} \sigma_n \left(E_{n,o} + E_{n,c}\right) + \sum_{n=1}^{N}\sum_{m=1}^{M} \mu_{n,m}\beta_{n,m} \\ & + \sigma_{n,c}\tilde{f}_n(t)\left(1 - \sum_{u \in \mathcal{N}\cup\mathcal{M}} \beta_{n,u}\right), \end{aligned} \tag{30}$$

where $\sigma_n = V\psi_n$ and $\sigma_{n,c} = V\psi_c$ are constants. For each TD, the cardinality of the candidate set is $M+N$, and the overall scheduling decision space is $(M+N)^n$. However, constrained by the service availability, communication ranges, delay requirements, and resource capacity, several candidates are not available. Hence, candidate pruning is necessary to accelerate the task scheduling in the initialization phase.

Remark 1. *For each TD n, all computing nodes i in the following set $\mathcal{F}_n$ is feasible for task scheduling*

$$\mathcal{F}_n = \begin{cases} f_{n,i}^{min} \le F_i \text{ and } \tau_{K_n} \ge \frac{d_{K_n}}{r_{n,i}^{max}}, & i \in \Omega_n \cup \{n\}, \\ f_{n,i}^{min} \le F_i \text{ and } \tau_{K_n} \ge D_{m,n,t}^{min}, & i \in \Omega_m, \end{cases} \tag{31}$$

where $f_{n,i}^{min}$ is the minimum computing resources required by TD n and varies among computing modes. $r_{n,i}^{max}$ is the maximum transmission rate of TD n to the helper TD i. $D_{m,n,t}^{min}$ is the minimum transmission delay for TD n to the ES m.

Proof. The detailed proof is shown in Appendix D. □

Nevertheless, obtaining the optimal solution via exhaustive search is still impractical due to the unacceptable computational complexity. To find an effective and low-complexity solution, we model task scheduling as a many-to-one matching problem, where each TD is scheduled to at most one node, and each node can serve many TDs. However, different from the traditional marriage matching problem where each player has a strict preference list over the players in the opposite set, the resource sharing on computing nodes causes the dynamic preference order to vary with the matching state, which makes the matching problem with peer effects becomes more complex [34]. Although the swap matching game for pursuing a two-side exchange-stable matching state was proposed to cope with the dilemma, the corresponding computation complexity will significantly increase due to the frequent and redundant swap operations and resource allocation optimization nested in it. To this end, we propose a heuristic matching-based method.

First, we define the total cost consumption of each TD as the utility function to measure the preference of the TDs over the nodes in the opposite set, which is given by

$$UF_n(\Xi) = \begin{cases} \sigma_n p_n D_{n,u,o}^{D2D} + \sigma_u \kappa f_{u,n}^2 c_{K_n}(t), & u \in \mathcal{N}, \\ \sigma_n p_n D_{n,u,o}^{MEC} + \mu_{n,u}, & u \in \mathcal{M}, \\ \sigma_{n,c}\tilde{f}_n, & u = \varnothing. \end{cases} \tag{32}$$

Ξ is a many-to-one matching function from the set $\mathcal{N}$ to the unordered set of $\mathcal{M} \cup \mathcal{N} \cup \{\varnothing\}$, where $\Xi(n) = u$ means TD n is matched to the node u, and $\Xi(n) = \varnothing$ means TD n will be scheduled to the center cloud. To decrease the overall system cost, the preference list for each TD should be constructed by sorting utilities in $\mathcal{F}_n$ according to the following properties:

$$u \succ_n u' \Leftrightarrow UF_n(u, \Phi) < UF_n(u', \Xi'), \tag{33}$$

which indicates that TD n prefers $u = \Xi(n)$ than $u' = \Xi'(n)$.

Next, we discuss the process of the initialization and update of preference lists as outlined below.

*Initialization (line 4-7 of **Algorithm 2**):* Given the high cloud outsourcing cost, we aim to accomplish more tasks in edge layer and device layer. Therefore, we adopt the lower bound of computing resources, i.e. $f_{n,u}^{min}$ in **Remark 1**, for preference calculation. It can be represented as

$$f_{n,u} = \begin{cases} \frac{c_{K_n}}{\tau_{K_n}}, & u \in \mathcal{F}_n \cap \{n\}, \\ \frac{c_{K_n}}{\tau_{K_n} - \frac{d_{K_n}}{r_{n,i}^{max}}}, & u \in \mathcal{F}_n \cap \mathcal{N} \setminus \{n\}, \\ \frac{c_{K_n}}{\tau_{K_n} - D_{m,n,t}^{min}}, & u \in \mathcal{F}_n \cap \mathcal{M}, \end{cases} \tag{34}$$

where $r_{n,i}^{max} = B_{n,i}\log_2\left(1 + \frac{P_n h_{n,i}}{\sigma^2}\right)$ and $D_{m,n,t}^{min} = \frac{d_{K_n}}{r_{n,m_n}^{max}} + D_{n,i,b}^{MEC}$. Then, the utility can be obtained according to (32), and the preference lists for all TDs can be constructed.

*Update (line 9-13 of **Algorithm 2**):* For cost minimization, each TD should be scheduled to the node at the top of its preference list. However, the computing resources of TDs that match with the same node are coupled with each other. Hence, TDs should be scheduled in a judicious order to decrease the total system cost and cope with the peer effects. For this purpose, we develop two ordering schemes for task scheduling.

- Lowest Cost First (LCF): Aiming to minimize energy consumption on TDs, the TD with the lowest utility among all preference lists will always be scheduled first.
- Shortest Feasible set First (SFF): Note that the tasks with large transmission data size, CPU requirements, and strict delay requirements require more resources, thus enabling limited feasible sets and undertaking the risk of high outsourcing costs. Therefore, scheduling the TD with the shortest feasible set can effectively reduce cloud outsourcing costs. For the TDs with the same cardinality of the feasible set, the TD with the lowest cost consumption will be scheduled preferentially.

After selecting the scheduled TD n and the desired node u, the matching state will be updated by $\Xi(n) = u$ and TD n will

Algorithm 2 Matching game-based bilevel optimization

Input: The current Lagrangian multiplier $\boldsymbol{\mu}$ in $\widetilde{\textbf{(P3)}}$.
Output: The current task scheduling $\boldsymbol{\beta}$ and resource allocation solution $\boldsymbol{p}, \boldsymbol{f}$.

$\Xi(n) = \varnothing, \forall n \in \mathcal{N}$, unmatched TD set $\mathcal{N}_{um} = \mathcal{N}$.
repeat
  Initialization:
  for $n \in \mathcal{N}_{um}$ **do**
    Generate feasible candidate mode $\mathcal{F}_n$ based on (31).
    Calculate the utility for each feasible candidate and build preference list according to (32)-(34).
  end for
  Update:
  Remove the TDs with empty preference list from $\mathcal{N}_{um}$.
  Select the scheduled TD n according to LCF or SFF.
  Schedule TD n to the first node u in its preference list.
  Update matching state $\Xi(n) = u$ and remove the scheduled TD from $\mathcal{N}_{um}$.
  Update the remaining computing resources of u based on (35), **Theorem 2**, and **3**.
until $\mathcal{N}_{um} = \varnothing$
Convert the matching state Ξ into task scheduling $\boldsymbol{\beta}$, and obtain the optimal resource allocation $\mathbf{p}^*, \mathbf{f}^*$.

be removed from the unmatched TD set $\mathcal{N}_{um}$. Considering the TDs that matched with node u are coupled with each other, the resource allocation $f^*_{n,u}$ in the lower level can be obtained by the methods proposed in Section V-C2. At the j_{th} iteration, the remaining computing resources of u is updated as

$$F_u^j = F_u - \sum_{n \in \mathcal{U}_u} f^*_{n,u}, \tag{35}$$

Then, the feasible sets and preference lists of all remaining TDs should be updated according to (31)-(34). Repeating the steps mentioned above until the unmatched TD set $\mathcal{N}_{um}$ is empty, the task scheduling and resource allocation decisions can be obtained according to the final matching state Ξ.

For the heuristic matching game at the upper level, the complexity mainly lies in the preference list construction, which needs at most $N+M$ operations for each TD. Moreover, the maximum number of iterations needed for the matching is $N-1$. Hence, the complexity of the algorithm can be expressed as $\mathcal{O}_t = \mathcal{O}((N-1)(N+M))$.

2) Resource allocation in edge collaboration: According to the task scheduling $\boldsymbol{\beta}$, the TDs with $\beta_{n,u} = 1, u \in \mathcal{M}$, that adopt edge collaborative computing, can be further divided into M individual groups. Let $\mathcal{U}_m$ denote the set of TDs that offload their computing tasks to ES m. Based on the aforementioned parameters definition, the resource allocation optimization problem of the m_{th} group is formulated as

$$\textbf{(P4)}: \min_{\mathbf{p},\mathbf{f}} \sum_{n=1}^{|\mathcal{U}_m|} \sigma_n \frac{p_n d_{K_n}}{B_n \log_2\left(1 + \frac{p_n h_{n,m_n}}{\delta^2}\right)} \tag{36}$$

$$\text{s.t.} \quad (14d) - (14f), \tag{36a}$$

where $\sigma_n = V\psi_n$ is a constant and varies among TDs.

Lemma 2. *The optimal solution exists iff. constraint (14d) preserves equality.*

Proof. The objective function of problem **(P4)** is monotonically increasing with $\mathbf{p}$. On the premise of ensuring (14d) and (14f), the optimal solution can be obtained by gradually decreasing $\mathbf{p}$. Assume $(\mathbf{p}, \mathbf{f})$ as the optimal solution, which preserves the total delay strictly less than the threshold τ_{K_n}. Keep decreasing $\mathbf{p}$ until (14d) tends to be equal, the new power allocation guarantees all constraints, while making the function value smaller than $\mathbf{p}$. Therefore, it is a better solution than the optimal $(\mathbf{p}, \mathbf{f})$, which contradicts the assumption. Therefore, there always exists a better solution for any $(\mathbf{p}, \mathbf{f})$ that makes (14d) unequal. □

With **Lemma 2**, the optimal $f^*_{n,m}$ can be rewritten as

$$f^*_{n,m} = \frac{c_{K_n}(t)}{\tau_{K_n(t)} - D^{MEC}_{n,m,b} - D^{MEC}_{n,o}(p^*_n)}, \tag{37}$$

which is a decreasing function of p^*_n. Then, by substituting (37) into (14e) and removing (14d), the problem **(P4)** is reduced into the optimization of $\mathbf{p}$. According to the constraint (14f) and (37), the lower bound of the $f_{n,m}$ can be expressed as $\check{f}_{n,m} = f_{n,m}(P_n)$. Therefore, it can be inferred that problem **(P4)** has feasible solution iff $\sum_{n=1}^{|\mathcal{U}_m|} \check{f}_{n,m} \le F_m$, which guarantees all constraints are satisfied. In the following, we further discuss two cases.

Case 1: If $\sum_{n=1}^{|\mathcal{U}_m|} \check{f}_{n,m} = F_m$, the optimal solution of TD $n \in \mathcal{U}_m$ can be represented as $(P_n, \check{f}_{n,m})$.

Case 2: If $\sum_{n=1}^{|\mathcal{U}_m|} \check{f}_{n,m} < F_m$, we have the following lemma:

Lemma 3. *The optimal solution exists only when the constraint (14e) preserves equality.*

Proof. Assume $(\mathbf{p}, \mathbf{f})$ as the optimal solution when (14e) is at inequality and all constraints are met strictly. Increasing the allocated computing resources for some of the TDs until (14e) tends to be equal leading to lower power allocation. Hence, a better objective function value will be obtained. □

Based on the analyses above, **(P4)** can be reorganised as

$$\widetilde{\textbf{P4}}: \min_{\mathbf{p}} \sum_{n=1}^{|\mathcal{U}_m|} \sigma_n \frac{p_n d_{K_n}}{B_n \log_2\left(1 + \frac{p_n h_{n,m_n}}{\delta^2}\right)} \tag{38}$$

$$\text{s.t.} \quad \sum_{n=1}^{|\mathcal{U}_m|} f_{n,m}(p_n) = F_m, \tag{38a}$$

$$0 \le p_n \le P_n, n \in \mathcal{U}_m. \tag{38b}$$

Theorem 2. *By leveraging the Karush-Kuhn-Tucker (KKT) conditions for the problem* $\widetilde{\textbf{P4}}$*, the close-formed local optimal solution can be derived as*

$$p^*_n = \min\left\{\max\left\{\zeta_n^{-1}(\lambda^*_m/\sigma_n), 0\right\}, P_n\right\}, \tag{39}$$

where λ^*_m *is an unique optimum that satisfies*

$$\sum_{n=1}^{|\mathcal{U}_m|} f_{n,m}(p^*_n) = F_m. \tag{40}$$

Proof. The detailed proof and the expression of ζ_n are shown in Appendix E. □

According to **Theorem 2**, the optimal λ_m^* can be efficiently obtained from $\lambda_m \in (0, \hat{\lambda})$ by leveraging a bisection search, where $\hat{\lambda}$ is a sufficiently large value. Given λ_m^*, the optimal power allocation can be directly obtained by calculating (39). To achieve the precision threshold T_{th2}, the method needs $\mathcal{O}\left(\log_2(\frac{\hat{\lambda}}{T_{th2}})\right)$ number of iterations to be coverage. Within each iteration, the computational complexity can be represented as $\mathcal{O}(|\mathcal{U}_m|)$, which mainly lies in evaluating (40). Therefore, the complexity of solving **(P4)** is less than $\mathcal{O}_e = \mathcal{O}\left(MN\left(\log_2(\frac{\hat{\lambda}}{T_{th2}})\right)\right)$.

3) Resource allocation in device collaboration: Given task scheduling decision $\boldsymbol{\beta}$, the TDs with $\beta_{n,u} = 1, u \in \mathcal{N}$ adopt device collaborative computing. All TDs can be divided into N individual groups served by different TDs, and each of them is constituted by the TDs with a specific type of service request. Denote $\mathcal{U}_i$ as the TD set served by the helper TD i, and the resource allocation optimization problem in the i_{th} group is formulated as

$$\textbf{(P5)}: \min_{\mathbf{p},\mathbf{f}} \sum_{n=1}^{|\mathcal{U}_i|} \sigma_n \frac{p_n d_{K_n}}{B_n \log_2\left(1+\frac{p_n h_{n,i}}{\delta^2}\right)} + \sigma_i \kappa f_{n,i}^2 c_{K_n} \quad (41)$$

$$\text{s.t.} \quad (14d)-(14f). \quad (41a)$$

Lemma 4. *The optimal solution exists iff. constraint (14d) preserves equality.*

Proof. The objective function (41) is monotonically increasing with p_n and $f_{n,i}$. Assume $(\mathbf{p}, \mathbf{f})$ as the optimal solution to preserve the total delay strictly less than the threshold τ_{K_n}. Decreasing the value of any variable while ensuring all constraints are met will obtain a lower objective function value. Therefore, there always exists a better solution for any $(\mathbf{p}, \mathbf{f})$ that makes (41a) unequal. □

With **Lemma 4**, the optimal resource allocation for n with $\beta_{n,n} = 1$ can be derived as $p_n^* = 0, f_{n,i}^* = \frac{c_{K_n}}{\tau_{K_n}}$, and the power allocation for $\forall n \in \mathcal{U}_i \setminus \{i\}$ can be expressed as

$$p_n^* = \frac{\delta^2}{h_{n,i}}\left(2^{\frac{d_{K_n}}{v_n B_n}} - 1\right), \quad (42)$$

where $v_n = \tau_{K_n} - \frac{c_{K_n}}{f_{n,i}}$ is a concave function of $f_{n,i}$. By substituting p_n into problem **(P5)**, the original problem is reduced into the following form:

$$\widetilde{\textbf{(P5)}}: \min_{\mathbf{f}} \sum_{n=1}^{|\mathcal{U}_i|} \sigma_n v_n \frac{\delta^2}{h_{n,i}}\left(2^{\frac{d_{K_n}}{v_n B_n}} - 1\right) + \sigma_i \kappa f_{n,i}^2 c_{K_n} \quad (43)$$

$$\text{s.t.} \quad 0 < \sum_{n=1}^{|\mathcal{U}_i|} f_{n,i} \le F_i, \quad (43a)$$

$$\frac{\delta^2}{h_{n,i}}\left(2^{\frac{d_{K_n}}{v_n B_n}} - 1\right) \le P_n, n \in \mathcal{U}_i. \quad (43b)$$

Constraint (43b) can be further derived into

$$f_{n,i} \ge \frac{c_{K_n}}{\tau_{K_n} - D_{n,i,o}^{D2D}(P_n)}, \quad (44)$$

where $D_{n,i,o}^{D2D}(P_n) = \frac{d_{K_n}}{B_n \log_2(1+P_n h_{n,i}/\delta^2)}$.

Theorem 3. *Problem $\widetilde{\textbf{(P5)}}$ is a convex optimization problem.*

Proof. The detailed proof is shown in Appendix F. □

Based on **Theorem 3**, problem $\widetilde{\textbf{(P5)}}$ can be solved by adopting the interior point method as the CVX solver. According to [30] and [35], the complexity of the interior point method can be approximated by $\mathcal{O}(\sqrt{a}(a+b)b^2)$, where a is the number of inequality constraints, and b is the variable dimension. Let I denote the number of iterations required for solving each subproblem $\widetilde{\textbf{(P5)}}$, the complexity of solving (41) can thus be approximated by $\mathcal{O}_d = \mathcal{O}(NI\left(\sqrt{N+1}(2N+1)N^2\right))$.

D. Performance Analysis

1) Complexity analysis: The OJSTR algorithm with two nested loops is operated in an iteration manner. Specifically, complexity in the outer loop mainly lies in the update of the Lagrangian multipliers. According to (24), its computational complexity can be formulated as $\mathcal{O}(NM)$. In the inner loop, computational complexity mainly lies in the service placement optimization via dynamic programming method and joint task scheduling and resource allocation optimization by adopting a bilevel optimization framework. Based on the analysis aforementioned, the overall complexity can be approximated by $\mathcal{O}_s + \mathcal{O}_t + \mathcal{O}_d + \mathcal{O}_e$.

2) Optimality Analysis: Our proposed OJSTR can provide the following performance guarantee on time-averaged system cost and service placement cost queue backlogs:

Theorem 4. *Let the infimum time average cost performance with the overall information as Ψ^{opt}, the following conditions will always hold by adopting the OJSTR for any value of V:*

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\sum_{n=1}^{N} \mathbb{E}\left[\Psi_n(t)\right] \le \frac{B}{V} + \Psi^{opt}, \quad (45)$$

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[Q(t)\right] \le \frac{B + V\Psi^{opt}}{\epsilon}, \quad (46)$$

where $\epsilon > 0$ is a small value that represents the distance between the time-averaged service placement cost consumption by some control policy and the cost budget, and $B = \frac{1}{2}\left(C_{max}^2 + \widetilde{C}^2\right)$ is a finite constant.

Proof. The detailed proof is shown in Appendix G. □

VI. Numerical Results

In this section, we present simulations to verify the effectiveness of our algorithms and analysis proposed in the previous sections. We consider a hierarchical collaborative MEC system with a cloud, two ESs, and several TDs. Each ES has a radius of 200m [10], and the device communication radius is 30m [36]. All TDs are randomly distributed over the coverage region, and the services of interest may differ among time slots. Considering the service diversity, we set up 10 services with different multidimensional resource requirements, e.g., delay-sensitive, computation-intensive, and

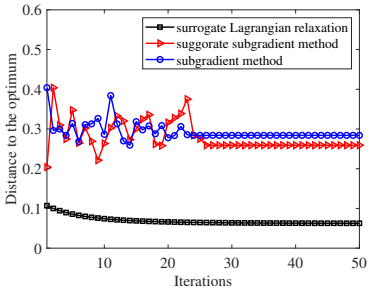


Fig. 3. Convergence performance of Algorithm 1

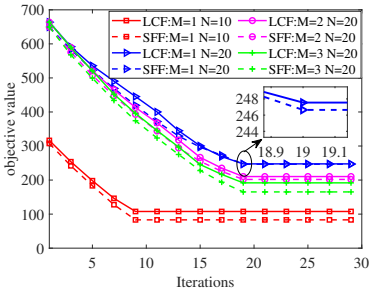


Fig. 4. Convergence performance of Algorithm 2

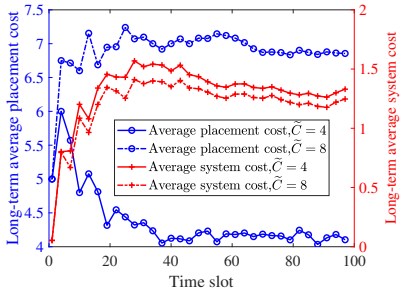


Fig. 5. Performance of OJSTR versus the service placement budget

TABLE II
SIMULATION PARAMETERS

Parameters	Value
Communication bandwidth W	2MHz [9]
Backhaul link bandwidth $w_{i,j}, i,j \in \mathcal{S}$	100Mbps [38]
Noise power density σ^2	-174dBm/HZ
Maximum transmit power of TDs P_n	0.1W [10]
Maximum storage capacity of ESs c	[10, 20]GB
Maximum CPU capacity of ESs F_i	[4, 10]GHz
Maximum CPU capacity of TDs F_i	[0.9, 1.5]GHz
Required storage of service requests	[2, 10]GB
Traffic data size of service requests	[0.2, 1]Mb
Required CPU cycles of service requests	$[1 * 10^7, 1 * 10^8]$cycles
Deadline service requests	[0.02, 0.05]s
Energy efficiency parameter κ	10^{-27}
Long term service placement cost budget $\widetilde{C}$	[4, 8]

storage-hungry services. The channel gain is assumed to remain constant within each time slot but may vary among time slots. Similar to [10], [37], the channel gain is modeled as $h_{n,u} = 127+30\times\log d_{n,u}$, where $d_{n,u}$ is the distance between the TD n and the computing node u. The other specific system parameters, including service requests and network settings, are listed in Table II.

For performance evaluation, we consider the following benchmarks. Moreover, the performance of all algorithms is evaluated by simulation implemented on the MATLAB platform, where the problem **(P5)** in OJSTR is addressed by the CVX package. All simulations are performed on a computer with a 2.50 GHz Core i7 CPU and 64 GB RAM.

- Global MEC Computing (GMEC): In this scheme, all service requests can only be executed at the network edge for satisfactory service provisioning. Otherwise, they will be scheduled to the cloud [23], [26].
- Global D2D Computing (GD2D): In this scheme, all service requests can only be computed locally or assisted by other TDs in the vicinity. Otherwise, cloud outsourcing will be adopted as the last resort [15].
- Alternating Optimization-based Solution (AOS): The alternating optimization framework is invoked to solve task scheduling and resource allocation subproblems iteratively, where the task scheduling subproblem with integer variables is solved by relaxing it as a linear programming (LP) problem [39].

Fig. 3 demonstrates the convergence behavior of the subgradient method, surrogate subgradient method, and our adopted surrogate Lagrangian relaxation method by comparing with the optimal value obtained by the exhaustive search method. For a fair comparison, we adopt the same objective function value initialized randomly to update stepsizes considering the unavailability of the optimal dual value in practice. In addition, all subproblems are fully optimized per iteration in the subgradient method. In contrast, only one subproblem is solved at a time to update multipliers in the other two surrogate subgradient-based methods. Therefore, the required computational effort of the subgradient method increases multiple times compared with others during the convergence process. In Fig. 3, we observe that the surrogate Lagrangian relaxation method achieves superior convergence speed and accuracy performance. Conversely, the other two methods exhibit obvious zigzagging during the iteration process and obtain a relatively worse objective function value at the end. The reason mainly lies in that our initialized objective value may be larger than the optimal, thus causing the upper bound of stepsize to violate condition (9) in [31] and leading to divergence. Conversely, the surrogate Lagrangian relaxation method adopts a novel stepsizing design as in (25) to guarantee convergence without requiring the optimal dual value.

Fig. 4 illustrates the convergence performance of **Algorithm 2** with different task ordering schemes, i.e., our proposed LCF and SFF. For a comprehensive evaluation, we randomly generate 10-time slots with varying system parameters, including the number of BSs and TDs. It can be seen that the number of iterations to achieve convergence will not exceed the number of TDs, thus indicating its linear convergence even in large networks. In addition, we observe that the objective function value of **(P3)** increases significantly when the number of TDs increases due to more energy consumption and cloud outsourcing costs for service provisioning. Benefiting from sufficient service placement and enlarged computing capabilities in horizontal edge collaboration, the convergence value decreases as the number of ESs increases as expected. We also observe that the task scheduling with SFF scheme always performs better than LCF, especially when the network resources are relatively scarce. This is because SFF emphasizes minimizing the outsourcing costs alongside taking into account the energy costs of TDs. Conversely, LCF only focuses on the TDs with the lowest energy costs, thereby leading to resource-hungry tasks being scheduled to the cloud with a higher probability.

Fig. 5 shows the long-term average service placement cost and system cost of OJSTR with various lengths of time slots

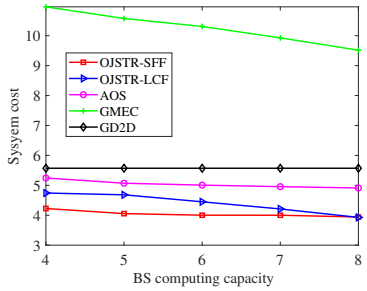


Fig. 6. Performance comparisons versus the threshold of ES computing capacity

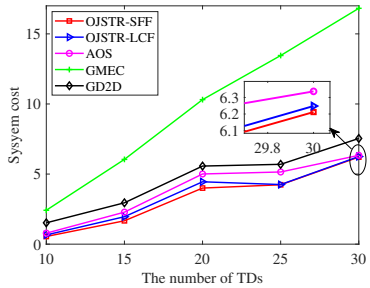


Fig. 7. Performance comparisons versus the number of TDs

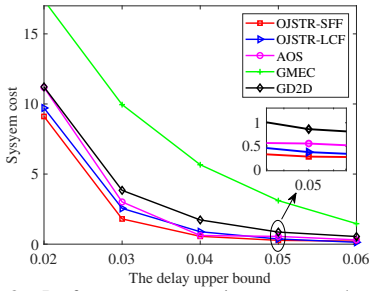


Fig. 8. Performance comparisons versus the delay upper bound of service requests

under the different setting of service placement cost budgets, which are defined as $\frac{\sum_{i=1}^{t}\sum_{m=1}^{M} C_m(i)}{t}$ and $\frac{\sum_{i=1}^{t}\sum_{n=1}^{N} \Psi_n(i)}{t}$, respectively. It can be observed that the varying curves of the average service placement cost gradually become stable, and the long-term cost budget can always be satisfied. This demonstrates the effectiveness of OJSTR in decreasing the total system cost as well as stabilizing queue backlogs over the long run. We also observe that the system cost can be significantly reduced with the service placement budget increasing when queue stability is ensured. This can be explained by the fact that the popular services are enabled to be deployed on demand when the service placement budget is large, thereby providing service for more TDs in a lower cost consumption.

Fig. 6 illustrates the impact of computing resource constraints of ESs on the performance of system cost. It is easy to see that the total cost consumption of TDs decreases with an increase in the MEC computing resources except GD2D, and GMEC exhibits the most significant reduction. This is because the dependence on edge resources for these schemes i.e., GMEC, HCC, and GD2D, decreases sequentially, and enlarging the computing power of ESs will provide more superior scheduling candidates for schemes with high dependence. Moreover, our proposed OJSTR with SFF and LCF task ordering schemes achieves superior performance compared to other schemes, which verifies the effectiveness of the HCC framework. Meanwhile, our proposed OJSTR can obtain proper scheduling decisions to unleash the collaboration space's service provisioning capability fully. By observing that the SSF scheme reaches a great performance when F_m = 5HZ, while LCF requires F_m = 8HZ to achieve relatively equivalent performance, we conclude that SSF enables to achieve excellent system performance with constrained network resources.

Fig. 7 shows the system cost performance of different algorithms when the number of TDs varies from 10 to 30. Obviously, the system cost increases with the number of TDs for all schemes because more energy and outsourcing costs are consumed for satisfactory service provisioning. It can also be seen that the system cost of GMEC is rapidly increasing due to the limited computing resources and increasing computational load. However, for other algorithms that invoke device collaboration, an increase in TDs also provides more idle computational capacity, thereby effectively alleviating the network computing load. Overall, the algorithms within HCC framework achieve better performance than GMEC and GD2D. Moreover, the bilevel optimization framework with heuristic matching game outperforms the AO-based method, i.e., the cost of SSF and LCF schedule schemes is reduced by an average of 17% and 16.5% compared to AOS, respectively.

Fig. 8 shows the system cost under different settings of delay upper bound of service requests. It is evident that the total system cost of all algorithms exhibits a substantial decrease when the delay deadline is relaxed while the task workload and total resources remain unchanged. This is because the feasible scheduling range for all tasks expands, and HCC enhances the resource utilization of different network layers. Moreover, our proposed OJSTR-SFF consistently outperforms the others. However, all schemes except GMEC demonstrate similar performance when the delay upper bound exceeds 0.05s. This is because the limited storage resources and the service placement budget restrict the deployment of services on ESs, resulting in the underutilization of computing resources in the edge collaboration plane. In contrast, HCC can further utilize the service programs and idle resources in proximate devices, thereby increasing the success rate of scheduling and reducing overall system costs.

Fig. 9 plots the distribution of device energy consumption and cloud tenancy cost versus the number of BSs. Here, the device energy consumption of GMEC under $M = 2$ and $M = 3$ are 0.0139 and 0.0205, respectively. According to the cost distribution of GMEC, increasing BS significantly reduces cloud tenancy costs due to the improvement of the system's service provisioning ability. Moreover, it is obvious that significant cost savings for TDs in both energy consumption and cloud tenancy can be achieved by increasing the number of BSs due to the tiny energy consumption for workload offloading in edge computing mode. We also observe that OJSTRA-SFF and OJSTRA-LCF can always achieve the lowest cloud tenancy cost and device energy consumption among all algorithms, respectively. This is mainly because the SFF scheme preferentially schedules the TDs with the shortest feasible set, while the LCF scheme pays more attention to the TDs with the lowest energy consumption.

Fig. 10 shows the distribution of all requests on different layers under different numbers of TDs and service request data sizes. This helps us understand the importance of different computing modes when our proposed OJSTR is used. The data clearly shows that the proportion of computation at the device layer increases with both an increase in the number of TDs and offloaded data volume. This means that device collaboration is important in meeting service demands when network loads

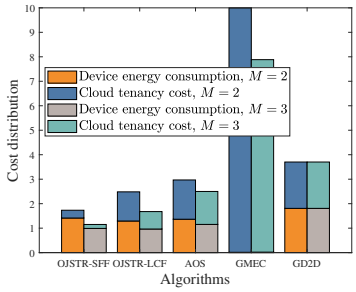


Fig. 9. Distribution of device energy consumption and cloud tenancy cost versus the number of BSs

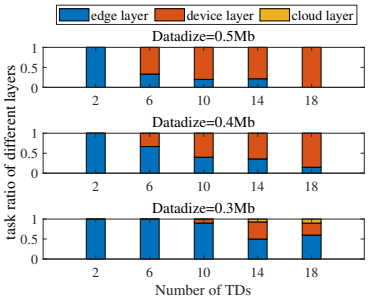


Fig. 10. Distribution of requests processing in different layers versus the number of TDs and requests data size

increase. On the one hand, though edge collaboration provides computing resources at a lower cost, these resources become scarce as the number of TDs increases. On the other hand, the consumed cost and delay for edge computing mainly depend on the data transmission process, thus a heavier transmission load leads to poor service quality. Benefiting from the idle computing resources and communication efficiency of device collaboration, its powerful potential will be fully unleashed as the network load increases. Therefore, our proposed OJSTR can provide TDs with satisfactory services at lower costs in different network scenarios by effectively leveraging the advantages of the HCC framework.

VII. Conclusion

In this paper, we proposed a novel HCC framework, which adopts horizontal collaboration in both the edge and device layer and vertical end-edge-cloud collaboration for serving spatially and temporally changing service demand patterns. To achieve heterogeneity-aware distributed resource management and uniform task scheduling cost-efficiently, we investigated a long-term cost consumption minimization problem by joint service placement, task scheduling, uplink transmission power, and computational resources allocation, subject to the long-term service placement cost budget, multidimensional resource constraints, and diverse task deadlines. Given the inability to predict future network information, we proposed an online optimization framework for problem transformation and solution derivation, in which the Lyapunov optimization theory was adopted to turn the problem into a pure queue stability problem, and a surrogate Lagrangian relaxation-based two-loop optimization framework was developed for solving the per-slot problems. Simulation results revealed that our proposed algorithms have great convergence performance, and the proposed OJSTR can efficiently manage network resources in the cooperative space to reduce the total system cost while ensuring the service placement budget.

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 2017.

[2] F. Spinelli and V. Mancuso, "Toward enabled industrial verticals in 5g: A survey on mec-based approaches to provisioning and flexibility," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 596–630, 2021.

[3] Y. Sahni, J. Cao, L. Yang, and Y. Ji, "Multi-hop multi-task partial computation offloading in collaborative edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1133–1145, 2021.

[4] C. Feng, Q. Yang, T. Q. Quek, W. Wu, and K. Guo, "Spatially-temporally collaborative service placement and task scheduling in mec networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 12, pp. 16 650–16 666, 2023.

[5] L. U. Khan, I. Yaqoob, N. H. Tran, S. M. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge-computing-enabled smart cities: A comprehensive survey," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10 200–10 232, 2020.

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

[7] 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 Trans. Mobile Comput.*, vol. 21, no. 10, pp. 3519–3535, 2022.

[8] T. Fang, F. Yuan, L. Ao, and J. Chen, "Joint task offloading, d2d pairing, and resource allocation in device-enhanced mec: A potential game approach," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3226–3237, 2022.

[9] Y. Pan, C. Pan, K. Wang, H. Zhu, and J. Wang, "Cost minimization for cooperative computation framework in mec networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3670–3684, 2021.

[10] Z. Xiao, J. Shu, H. Jiang, J. C. S. Lui, G. Min, J. Liu, and S. Dustdar, "Multi-objective parallel task offloading and content caching in d2d-aided mec networks," *IEEE Trans. Mobile Comput.*, pp. 1–16, 2022.

[11] H. Xiao, J. Huang, Z. Hu, M. Zheng, and K. Li, "Collaborative cloud-edge-end task offloading in mec-based small cell networks with distributed wireless backhaul," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 4, pp. 4542–4557, 2023.

[12] Y. Cui, V. K. N. Lau, R. Wang, H. Huang, and S. Zhang, "A survey on delay-aware resource control for wireless systems—large deviation theory, stochastic lyapunov drift, and distributed stochastic learning," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1677–1701, 2012.

[13] 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 Trans. Mobile Comput.*, vol. 22, no. 7, pp. 4150–4167, 2023.

[14] X. Jiang, F. R. Yu, T. Song, and V. C. M. Leung, "A survey on multi-access edge computing applied to video streaming: Some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 871–903, 2021.

[15] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 360–374, 2021.

[16] J. Peng, H. Qiu, J. Cai, W. Xu, and J. Wang, "D2d-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for mec," *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 4858–4873, 2021.

[17] X. Wang, Y. Han, H. Shi, and Z. Qian, "Joagt: Latency-oriented joint optimization of computation offloading and resource allocation in d2d-assisted mec system," *IEEE Wireless Commun. Lett.*, vol. 11, no. 9, pp. 1780–1784, 2022.

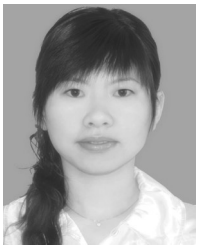
[18] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and resource allocation for d2d-enabled mobile-edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4193–4207, 2019.

[19] 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 Trans. Mobile Comput.*, vol. 23, no. 1, pp. 238–256, 2024.

[20] Z. Teng, J. Fang, and Y. Liu, "Combining lyapunov optimization and deep reinforcement learning for d2d assisted heterogeneous collaborative edge caching," *IEEE Trans. Netw. Service Manag.*, pp. 1–1, 2024.
[21] Y. Zhao, A. Xiao, S. Wu, C. Jiang, L. Kuang, and Y. Shi, "Adaptive partitioning and placement for two-layer collaborative caching in mobile edge computing networks," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2024.
[22] X. Li, X. Zhang, and T. Huang, "Joint task offloading and service placement for mobile edge computing: An online two-timescale approach," *IEEE Trans. on Cloud Comput.*, vol. 11, no. 4, pp. 3656–3671, 2023.
[23] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Service placement and request routing in mec networks with storage, computation, and communication constraints," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1047–1060, 2020.
[24] J. Li, W. Liang, M. Chen, and Z. Xu, "Mobility-aware dynamic service placement in d2d-assisted mec environments," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, 2021, pp. 1–6.
[25] K. Li, X. Wang, Q. He, M. Yang, M. Huang, and S. Dustdar, "Task computation offloading for multi-access edge computing via attention communication deep reinforcement learning," *IEEE Trans. Serv. Comput.*, vol. 16, no. 4, pp. 2985–2999, 2023.
[26] X. Chen, Y. Bi, X. Chen, H. Zhao, N. Cheng, F. Li, and W. Cheng, "Dynamic service migration and request routing for microservice in multicell mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13 126–13 143, 2022.
[27] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao, "Task offloading with network function requirements in a mobile edge-cloud network," *IEEE Trans. Mobile Comput.*, vol. 18, no. 11, pp. 2672–2685, 2019.
[28] X. Ma, S. Zhang, P. Yang, N. Zhang, C. Lin, and X. Shen, "Cost-efficient resource provisioning in cloud assisted mobile edge computing," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6.
[29] S. Bi, L. Huang, H. Wang, and Y.-J. A. Zhang, "Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7519–7537, 2021.
[30] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge Univ. Press, 2004.
[31] M. A. Bragin, P. B. Luh, J. H. Yan, N. Yu, and G. A. Stern, "Convergence of the surrogate lagrangian relaxation method," *Journal of Optimization Theory and applications*, vol. 164, pp. 173–201, 2015.
[32] X. Zhao, P. Luh, and J. Wang, "The surrogate gradient algorithm for lagrangian relaxation method," in *Proceedings of the 36th IEEE Conference on Decision and Control*, vol. 1, 1997, pp. 305–310 vol.1.
[33] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 276–295, 2018.
[34] W. He, D. He, X. Ma, X. Chen, Y. Fang, and W. Zhang, "Joint user association, resource allocation, and beamforming in ris-assisted multi-server mec systems," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2023.
[35] F. Jarre, "Interior-point methods for convex programming," *Appl Math Optim*, vol. 26, no. 3, pp. 287–311, Nov. 1992. [Online]. Available: https://doi.org/10.1007/BF01371086
[36] X. Dai, Z. Xiao, H. Jiang, M. Alazab, J. C. S. Lui, S. Dustdar, and J. Liu, "Task co-offloading for d2d-assisted mobile edge computing in industrial internet of things," *IEEE Trans. Ind. Informat.*, vol. 19, no. 1, pp. 480–490, 2023.
[37] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 939–951, 2021.
[38] Q. Yuan, J. Li, H. Zhou, T. Lin, G. Luo, and X. Shen, "A joint service migration and mobility optimization approach for vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 9041–9052, 2020.
[39] Y. Qu, H. Dai, H. Wang, C. Dong, F. Wu, S. Guo, and Q. Wu, "Service provisioning for uav-enabled mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 11, pp. 3287–3305, 2021.

VIII. Biography

An Du received the B.S. degree in computer science and technology from Taiyuan Normal University, China, in 2019, and the M.S. degree in computer technology from Northeastern University, Shenyang, China, in 2021, where she is currently pursuing the Ph.D. degree in computer science and technology. Her research interests include edge computing, edge intelligence, machine learning, and network function virtualization.



Jie Jia (Member, IEEE) received the Ph.D. degree in Computer System Architecture from Northeastern University, China, in 2009. She is currently working as a Professor with the School of Computer Science and Engineering, Northeastern University. In 2016, she worked as a Visiting Research Associate with the Department of Informatics, King's College London (KCL). She has published over 100 technical papers on various aspects of wireless networks. Her current research mainly focuses on HetNets, IoT, and cognitive radio networks. She is a member of various international societies, such as China Computer Federation (CCF).

Schahram Dustdar (Fellow, IEEE) is a Full Professor of Computer Science (Informatics) with a focus on Internet Technologies heading the Distributed Systems Group at the TU Wien. He is founding co-Editor-in-Chief of ACM Transactions on Internet of Things (ACM TIoT) as well as Editor-in-Chief of Computing (Springer). He is an Associate Editor of IEEE Transactions on Services Computing, IEEE Transactions on Cloud Computing, ACM Computing Surveys, ACM Transactions on the Web, and ACM Transactions on Internet Technology, as well as on the editorial board of IEEE Internet Computing and IEEE Computer. He is recipient of multiple awards: TCI Distinguished Service Award (2021), IEEE TCSVC Outstanding Leadership Award (2018), IEEE TCSC Award for Excellence in Scalable Computing (2019), ACM Distinguished Scientist (2009), ACM Distinguished Speaker (2021), IBM Faculty Award (2012).

Jian Chen (Member, IEEE) received the Ph.D. degree in Computer Application Technology from Northeastern University in 2010. He is currently working as an Professor with the School of Computer Science and Engineering, Northeastern University. In 2016, he worked as a Visiting Research Associate with the Department of Informatics, King's College London (KCL). His research interests include intelligent reconfigurable surface (IRS), D2D communication, location technology, network management, and signal and image processing.

Xingwei Wang received the BS, MS, and PhD degrees in computer science from the Northeastern University, Shenyang, China, in 1989, 1992, and 1998, respectively. He is currently a professor with the College of Computer Science and Engineering, Northeastern University, Shenyang, China. His research interests include cloud computing and future Internet, etc. He has published more than 100 journal articles, books and book chapters, and refereed conference papers. He has received several best paper awards.