EdgeOPT: A Competitive Algorithm for Online Parallel Task Scheduling With Latency Guarantee in Mobile Edge Computing

Yuchen Yang and Shaowei Wang, *Senior Member, IEEE*

***Abstract*— The paradigm of mobile edge computing (MEC) has emerged as a promising solution to the increasing demand of time-sensitive applications, where tasks generated by users are offloaded to proximate edge clouds for low-latency execution. Due to the online nature of task generation and edge capacity bottleneck, a fundamental challenge for the MEC network is how to optimally schedule the tasks and resources in face of uncertain future arrivals. To this end, we propose a competitive algorithm named *EdgeOPT* for online parallel task scheduling aiming to maximize the cumulative reward of completed tasks subject to their hard deadlines. The algorithm leverages an adaptive threshold structure at each server to schedule tasks with different demand patterns based on the status of the system and all active users, while incorporating a subroutine for efficient resource allocations. We prove a bounded competitive ratio for *EdgeOPT* when scheduling monolithic tasks, and propose its extended version to schedule chains of dependent functions. We conduct extensive experiments to demonstrate the effectiveness and superiority of our proposal compared to all the baselines.**

***Index Terms*— Mobile edge computing, online combinatorial optimization, resource management.**

I. Introduction

THE proliferation of computing-intensive applications is propelling the telecommunication industry towards embracing the paradigm of mobile edge computing (MEC), where low-latency computational services are provided by servers at the network edge [1]. By harnessing network function virtualization and software-defined networks, the agile and scalable service provisioning of edge clouds holds the potential to greatly enhance the capabilities of mobile devices while simultaneously mitigating the burdens on radio and back-haul networks brought by the remote cloud [2], [3], [4], [5], making MEC highly promising in both academia and industry.

In an MEC network, a fundamental challenge is the design of offloading-scheduling mechanism, which determines the user-edge association and the allocation of multiple resources. Much effort has been devoted to this problem in offline settings, where the formulated NP-hard problems are optimized with respect to various metrics quantifying the quality of service [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16]. For instance, task offloading is investigated under software-defined networks, where heuristic methods are introduced to minimize the sum task duration [6] and the cost of delay and energy consumption [12]. Offloading and scheduling are jointly studied in [9], where the number of admitted tasks is maximized by simultaneously optimizing the offloading decision, executing order and computing resource allocation. Network slicing is taken into consideration in [16], where an approximation algorithm is devised to minimize the sum completion time of computational tasks by jointly optimizing the offloading decisions and the edge resource sharing policy both within and across slices.

While the aforementioned formulations exhibit clarity and simplicity, they nonetheless suffer from critical defects. Firstly, the offline approaches heavily rely on the global information of concurrently generated tasks, contrasting with the online nature of task generation in practical systems. The complexity introduced by the offline centralized algorithms can scarcely meet the requirement of real-time responsiveness, which hinders their practical implementations. In addition, the need for a central controller renders offline schemes unscalable due to their super-linearly growing signaling overhead and execution delay when applied to large-scale systems. Furthermore, the heterogeneity of task completion times necessitates resource redistribution over time, whereas offline models are agnostic to this spatial and temporal correlation among users, leading to highly suboptimal solutions. To better capture the system dynamics, several works employ online models, in which online offloading decisions are produced through online learning [17], [18], [19], reinforcement learning [20], [21], [22], and Lyapunov optimization [23]. In practice, however, these user-initiated schemes suffer from unrevealed system-side information [17], which impedes the optimal system operation. It would be of great help if the offloading decisions can be scheduled by the edge clouds with clear system status at hand. This leads to the online scheduling problem, where computation tasks are generated online and the edge clouds need to manage multiple resources across time.

Even with full information of system status, the online nature of task generation and edge resource bottleneck can still make the scheduling problem significantly hard, as decisions can only be made in face of unknown future task arrival. Early

Manuscript received 23 August 2023; revised 1 January 2024 and 28 April 2024; accepted 2 June 2024. Date of publication 11 June 2024; date of current version 14 November 2024. This work was partially supported by the National Natural Science Foundation of China under Grants 61931023. The associate editor coordinating the review of this article and approving it for publication was M. Levorato. *(Corresponding author: Shaowei Wang.)*

The authors are with the School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China (e-mail: yuchenyang@smail.nju.edu.cn; wangsw@nju.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCOMM.2024.3412741.

Digital Object Identifier 10.1109/TCOMM.2024.3412741

works mostly focus on online scheduling in unary servers [24], [25], [26], [27], where only one task can be processed at a time and the system decides the offloading locations and execution orders for the offloaded tasks. In [24], a threshold preemptive algorithm is presented aiming to maximize the values of fully completed online tasks. In [25], the weighted response time of all preemptive tasks is minimized through an online approximation scheme under $(1+\epsilon)$-speed augmentation model [28]. With the same adoption, [27] targets on maximizing the total heterogenous utility functions of all online tasks. In [26], maximizing the number of tasks meeting their deadlines is handled by scheduling both the computing and networking resources. Parallel scheduling is considered in [29], where the resource requirement and processing time of each task is assumed to be fixed. However, the disregard for the dependence of execution delay on allocated resources implies that the system utility has not been fully exploited. Indeed, it has been shown that supporting flexible resource allocation while exploiting parallelism is critical for improving the execution time and energy consumption of the offloaded tasks [30]. Driven by the escalating demand of latency-bounded tasks [31], it becomes imperative to further exploit the potential of an edge computing system through integrating dynamic resource allocation into the scheduling of online parallel tasks.

In this work, we study the online parallel scheduling problem in an MEC network, where tasks with hard deadlines are generated in arbitrary order and times to be offloaded to a set of resource-constrained edge servers. Upon each task arrival, the servers need to schedule the optimal subset of task requests and manage multiple resources such that, within arbitrary time horizon, the cumulative reward of all completed tasks subject to their hard deadlines is maximized. As required by many mission-critical and emergency systems, our proposal provides a worst-case performance guarantee [25], [32] quantified by the metric of *competitive ratio*, which is defined as the maximum ratio between the offline optimum and the algorithm's output under every possible instance. Our contributions can be summarized as follows:

- We investigate the online parallel task scheduling problem for multiple edge clouds under hard deadline constraint. Taking into account both sharable and non-sharable resources, we formulate this problem into a general model that captures both the edge capacity limitation and the task state transitions among transmission, computation and completion. To the best of our knowledge, this is the first work to study the deadline-aware online scheduling problem with parallel task execution and dynamic resource allocation in multiple edge clouds.
- We propose *EdgeOPT*, a competitive algorithm for online parallel task scheduling with latency guarantees. Our proposal does not require any prior global knowledge, and is scalable to large-scale edge computing systems. In *EdgeOPT*, the scheduling decisions are optimized leveraging the adaptive thresholds at each server based on the status of the system and all active users. Meanwhile, it allows efficient resource allocation upon each task state transition through a subroutine which exploits the inherent water-filling structure of the optimization policy.

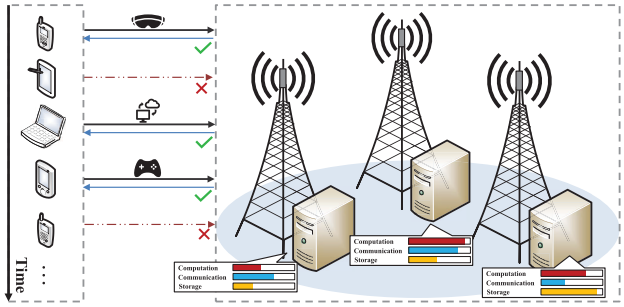

Fig. 1. Online task scheduling for multiple edge clouds.

- We theoretically analyze the worst-case performance and yield an $\mathcal{O}(\lambda\theta\ln(\theta))$ competitive ratio in the case of monolithic task scheduling, where λ and θ are the fluctuations of task deadline and resource intensity, respectively. We also extend *EdgeOPT* and propose an efficient algorithm to schedule chains of dependent virtual functions.
- We conduct extensive experiments under various network settings and task patterns, whose results demonstrate that our proposed algorithm significantly improves the cumulative reward compared with all the baselines, which confirms the theoretical analysis as well as the effectiveness of our proposal.

The rest of this paper is organized as follows. In Section II, the system model and problem formulation are described. In Section III, we present the algorithm and theoretically analyze its competitive ratio and time complexity. Numerical experiments under different settings are conducted and analyzed in Section IV. In Section V, we conclude the paper. We defer some of the proofs to the Appendices.

II. System Model and Problem Formulation

The system model is illustrated in Fig. 1. Consider an MEC system with a set $\mathcal{M}$ of edge clouds covering an unknown set $\mathcal{K}$ of users, each configured with a computation task. Each task is represented by a virtual function belonging to a specific type chosen from the set Π. Task generated by user $k \in \mathcal{K}$ can be characterized by a tuple $(s_k, w_k, l_k, \tau_k, \mathcal{R}_k)$, where s_k is the packet size for data transmission and w_k is the computation workload demand, which can be measured using methods in [33]. The function type of task k is specified by $l_k \in \Pi$, each type requires its dedicated container to provide the necessary environment and dependencies for execution. If a server does not provide such instance before execution, its image should be fetched and loaded, which incurs a time cost of z_{l_k} and consumes $\mathcal{D}_{l_k}$ memory. Accepted tasks must be completed within the maximum tolerable delay τ_k to receive reward $\mathcal{R}_k$.

The task requests are generated in arbitrary order and times, and are immediately delivered to the network. We adopt orthogonal frequency-division multiple access for uplink transmission, where the frequency band at each base station m is divided into a set of sub-channels with bandwidth W [15], [34]. Let $\mathcal{C}_m$ be the set of sub-channels at base station m, we define $\mathcal{C} := \bigcup_{m\in\mathcal{M}} \mathcal{C}_m$. Upon the online arrival of the

generated tasks, the edge servers need to schedule a subset of the revealed tasks such that the cumulative reward of tasks completed before their deadlines is maximized. To this end, we need to decide whether and where to accept the tasks upon their arrivals, select the transmission channels for all accepted tasks, and manage computation resources throughout their executions. If a task $k \in \mathcal{K}$ is accepted by a server $m \in \mathcal{M}$, the user uploads the task packet sized s_k with transmission power p_k and bandwidth W. Denote by r_{mkt} the achievable uplink data rate between user k and edge cloud m at time t, we have:

$$r_{mkt} = W \log_2 \left(1 + \frac{p_k |h_{mk}|^2}{\Gamma(N_0 W + I_{kmt})}\right), \tag{1}$$

where Γ represents the SNR gap for a given bit error rate under a specific modulation/demodulation scheme [35]. N_0 is the power spectral density of additive white Gaussian noise. For ease of analysis, and given that the offloading process generally operates on a timescale significantly larger than the channel coherence time, we assume the channel gain h_{mk} is constant to represent the average value during the process [7], [22], [36], [37], [38]. Let $\mathbb{1}_X$ be the indicator function which equals to 1 when X is TRUE and equals to 0 otherwise. At any time, the inter-cell interference power can be calculated as: $I_{kmt} = \sum_{m \in \mathcal{M}} \sum_{i \in \mathcal{S}^B_{t,m} \setminus k} |h_{mi}|^2 p_i \mathbb{1}_{c_i = c_k}$, where $c_k \in \mathcal{C}$ is the transmission channel of user k, and $\mathcal{S}^B_{t,m}$ is the set of users transmitting packets to $m \in \mathcal{M}$ at time t. We assume the maximum allowable interference power of user k is $\hat{I}_k$, representing the worst acceptable transmission quality specified by the service level agreement between user k and the service provider. Denote by $\hat{\mathfrak{r}}_{mk}$ the actual upload delay between user k and server m, and by $\underline{\mathfrak{r}}_{mk}$ the specified transmission delay estimated using the interference power $\hat{I}_k$. We further define $\mathcal{S}^C_{t,m}$ to be the set of tasks whose delays have not reached $\underline{\mathfrak{r}}_{mk}$.

Received tasks are processed using containerized or virtualized environments created by the server and are allocated with computation resources [1], [30], [39]. Denote by a_k the arrival time of task request from k and $\mathcal{F}_k(t)$ the allocated computation resource for task k at time t. Let $\hat{z}_{l_k,k} = z_{l_k}$ if container l_k need to be instantiated when request k arrives, and $\hat{z}_{l_k,k} = 0$ otherwise. It takes $v_{mk} = \max\{\hat{\mathfrak{r}}_{mk}, \hat{z}_{l_k,k}\}$ time for task k to be ready for execution. It therefore requires

$$\int_{a_k + v_{mk}}^{a_k + \tau_k} \mathcal{F}_k(t) dt \geq w_k \tag{2}$$

for the workload w_k to be completed before the deadline. Completed tasks release their occupied resources to enable resource redistribution for unfinished tasks. The delays of transmitting task requests and computation results are ignored due to their negligible packet sizes in practice [16], [22].

Let x_{kmc} be the indicator for the scheduling decisions where $x_{kmc} = 1$ implies the task from user k is accepted by server m and is transmitted using channel c, $x_{kmc} = 0$ otherwise. Early rejection of unsatisfiable tasks is proved beneficial since users can seek alternatives that completes the task on time [40]. The offline formulation of the problem[1] is given by:

$$\begin{aligned}
\max_{x_{kmc}, \mathcal{F}_k} \quad & \sum_{k \in \mathcal{K}} \mathcal{R}_k \sum_{m \in \mathcal{M}} \sum_{c \in \mathcal{C}_m} x_{kmc} \\
\text{s.t.} \quad & C_1: \int_{a_k + v_{mk}}^{a_k + \tau_k} \mathcal{F}_k(t') dt' \geq w_k \sum_{m \in \mathcal{M}} \sum_{c \in \mathcal{C}_m} x_{kmc}, \forall k, \\
& C_2: \sum_{k \in \mathcal{K}} x_{kmc} \mathbb{1}_{t \in [a_k, a_k + \hat{\mathfrak{r}}_{mk}]} \leq 1, \forall t, m, c \\
& C_3: \sum_{k \in \mathcal{K}} \sum_{c \in \mathcal{C}_m} \mathcal{F}_k(t) x_{kmc} \mathbb{1}_{t \in [a_k + v_{mk}, d_k]} \leq \mathcal{F}^m, \forall t, m, \\
& C_4: \sum_{l \in \Pi} \mathcal{D}_l \mathbb{1}_{\left[\sum_{c,k:l_k = l} x_{kmc} \mathbb{1}_{t \in [a_k, d_k]} > 0\right]} \leq \mathcal{D}^m, \forall t, m, \\
& C_5: \sum_{i \in \mathcal{K} \setminus k} \sum_{n \in \mathcal{M}} p_i |h_{mi}|^2 x_{inc} \mathbb{1}_{t \in [a_i, a_i + \hat{\mathfrak{r}}_{ni}]} \\
& \leq \hat{I}_k + \Lambda(1 - x_{kmc} \mathbb{1}_{t \in [a_k, a_k + \hat{\mathfrak{r}}_{mk}]}), \forall t, m, k, c, \\
& C_6: \sum_{m \in \mathcal{M}} \sum_{c \in \mathcal{C}_m} x_{kmc} \leq 1, \forall k, \\
& C_7: x_{kmc} \in \{0, 1\}, \mathcal{F}_k(t) \geq 0, \forall t, m, k, c,
\end{aligned} \tag{3}$$

where d_k is the departure time of task k. Constraint C_1 ensures that all accepted tasks are completed before the deadlines. C_2 indicates that a sub-channel can only be occupied by one user at the same time. C_3 implies that the sum of allocated computation resource cannot exceed the total computation resource. C_4 restricts the aggregate memory requirement, in which Λ is a sufficiently large number. Unlike bandwidth or computing resource that are allocated to a dedicated request, memory can be shared in that multiple virtual functions of the same type can be served concurrently by one container instance without extra memory consumption [41], [42], [43]. C_5 is the interference constraint that prevents unlimited data transmission that affect the viability of the offloading process. C_6 ensures each task can only be assigned to one edge server. The frequently used notations are summarized in Table I.

We use the widely adopted *competitive ratio* to theoretically evaluate the worst-case performance, which is defined as:

Definition 1: Let $\mathcal{A}(\mathtt{I})$ be the output of online algorithm $\mathcal{A}$ over instance $\mathtt{I}$, and $OPT(\mathtt{I})$ be the offline optimum with full knowledge of instance $\mathtt{I}$. The competitive ratio $c(\mathcal{A})$ of $\mathcal{A}$ in a maximization problem is given by:

$$c(\mathcal{A}) = \max_{\mathtt{I}} \left\{ \frac{OPT(\mathtt{I})}{\mathcal{A}(\mathtt{I})} \right\}.$$

Hence, a c-competitive algorithm achieves at least $1/c$ fraction of the optimum value in the worst case.

III. Proposed Algorithm

We present a constraint reduction technique and propose the adaptive threshold based algorithm incorporating a subroutine for resource allocation. The complexity and competitive ratio are analyzed in the case of scheduling monolithic tasks. We also extend our algorithm to schedule virtual function chains.

[1]We may also add capacity constraints or transmission number constraints to prevent decisions that transmit an unlimited quantity of data. In the theoretical analysis section we show that the algorithm is still valid as long as the constraints added are linear or submodular.

TABLE I
LIST OF MAIN NOTATIONS

Notations	Descriptions
$\mathcal{M}$	The set of edge clouds
$\mathcal{K}$	The unknown set of users, whose information is revealed only after the task generations
$\mathcal{C}$	The set of communication channels
Π	The set of possible container types
$\mathcal{R}_k$	Reward associated with the task from user k
s_k	Packet size of the offloaded task from user k
w_k	Computation workload demand of user k
l_k	Function type of user k
$\mathcal{D}_l$	Memory requirement of type-l container
z_l	Start-up time of type-l container instance
τ_k	Hard deadline of task k
W	Transmission bandwidth of a sub-channel
r_{mkt}	Transmission rate between server m and user k at t
h_{mk}	Channel gain between server m and user k
a_k	Arrival time of task k
$\mathcal{F}_k(t)$	Allocated computing resource for task k at time t
$w'_k(t)$	Remaining workload of task k at time t
$\mathcal{S}_t^m$	The set of tasks accepted by server m at time t
$\mathcal{S}_{t,m}^B$	The set of tasks during transmission to server m at time t
$\mathcal{J}$	The set of constraints
$q_j(\boldsymbol{\varphi}_k, t)$	the resource demand for task k in dimension j at t under decision $\boldsymbol{\varphi}_k$
$u_j(\mathcal{S}_t^m)$	The server usage occupied by tasks in $\mathcal{S}_t^m$
C_j^m	The capacity of resource j for server m
λ	The deadline fluctuation
θ	The resource intensity fluctuation

A. Constraint Reduction

Let the decision variables x_{kmc} be the indicators for the decision set $\mathcal{S}$, where the scheduling decision to transmit user k's data through channel c to server m is included in $\mathcal{S}$ if and only if $x_{kmc} = 1$. We have the following definition.

Definition 2: (Submodular Constraint): For a ground set V and $\forall \mathcal{S} \subseteq V$, a constraint $F(\mathcal{S}) \leq A$ is called submodular constraint if $F(\mathcal{S}) : 2^V \mapsto \mathbb{R}$ is a submodular function, i.e., for all $\mathcal{S} \subseteq \mathcal{S}' \subseteq V$ and every $k \in V \setminus \mathcal{S}'$, we have:

$$F(\mathcal{S} \cup k) - F(\mathcal{S}) \geq F(\mathcal{S}' \cup k) - F(\mathcal{S}'). \tag{4}$$

It is straightforward to verify that C_4 in (3) is submodular: adding a new user to a server serving requests in $\mathcal{S}$ consumes extra memory no less than adding it to a server with requests in $\mathcal{S}' \supseteq \mathcal{S}$, as there is more possibility for the users to share a common container when the server contains more requests. Note that C_2, C_4, C_5 and C_6 are all submodular constraints, as linear function is a special case of submodular function. The feasible domain $\mathcal{P}$ of (3) is complicated by the variational constraints C_1 and C_3. To make the problem tractable, we approximate the feasible region with a subset $\mathcal{I} \subseteq \mathcal{P}$ which can be sufficiently represented by multiple submodular constraints. To this end, define $f'_k(t)$ as the minimum required computation resource for task k to finish its remaining workload at time t before the deadline. For every accepted task, it follows that:

$$w_k - \int_{a_k+v_{mk}}^{t} \mathcal{F}_k(\tau)d\tau \leq \int_{t}^{a_k+\tau_k} f'_k(t)d\tau, \tag{5}$$

which gives:

$$f'_k(t) = \frac{1}{a_k + \tau_k - t}\left(w_k - \int_{a_k+v_{mk}}^{t} \mathcal{F}_k(\tau)d\tau\right). \tag{6}$$

The constraint reduction technique restricts the family of $\mathcal{F}_k(t)$ such that $\mathcal{F}_k(t) \geq f'_k(t)$ for every $t \in [a_k + v_{mk}, a_k + \tau_k]$. This gives a new constraint:

$$\mathcal{F}_k(t) \geq f'_k(t),\ \forall k \in \mathcal{S}_t^m, m \in \mathcal{M}, t \in [a_k + v_{mk}, a_k + \tau_k], \tag{7}$$

where $\mathcal{S}_t^m$ is the set of tasks accepted by server m at time t. This guarantees that task k is allocated at least $f'_k(t)$ resource to be completed using at most τ_k time. On the other hand, the sum of all minimum requirement $f'_k(t), \forall k \in \mathcal{S}_t^m$ must not exceed the budget $\mathcal{F}^m$, which gives:

$$\sum_{k \in \mathcal{S}_t^m} f'_k(t)\mathbb{1}_{t\in[a_k+v_{mk},d_k]} \leq \mathcal{F}^m,\ \forall t, \forall m \in \mathcal{M}. \tag{8}$$

Replacing constraint C_1, C_3 and C_5 by the stronger constraints (7) and (8) yields the reduced problem:

$$\begin{aligned}
\max_{x_{kmc},\mathcal{F}_k}\quad & \sum_{k\in\mathcal{K}} \mathcal{R}_k \sum_{m\in\mathcal{M}} \sum_{c\in\mathcal{C}_m} x_{kmc} \\
\text{s.t.}\quad & C_1: \sum_{k\in\mathcal{K}} x_{kmc}\mathbb{1}_{t\in[a_k,a_k+\hat{\mathfrak{r}}_{mk}]} \leq 1, \forall t, m, c \\
& C_2: \sum_{k\in\mathcal{K}} \sum_{c\in\mathcal{C}_m} f'_k(t)x_{kmc}\mathbb{1}_{t\in[a_k+v_{mk},d_k]} \leq \mathcal{F}^m, \forall t, m, \\
& C_3: \sum_{l\in\Pi} \mathcal{D}_l \mathbb{1}_{\left[\sum_{c,k:l_k=l} x_{kmc}\mathbb{1}_{t\in[a_k,d_k]}>0\right]} \leq \mathcal{D}^m, \forall t, m, \\
& C_4: \sum_{k\in\mathcal{K}\setminus i} \sum_{m\in\mathcal{M}} p_k|h_{nk}|^2 x_{kmc}\mathbb{1}_{t\in[a_k,a_k+\hat{\mathfrak{r}}_{mk}]} \\
& \leq \hat{I}_i + \Lambda(1 - x_{inc}\mathbb{1}_{t\in[a_i,a_i+\hat{\mathfrak{r}}_{ni}]}), \\
& \forall n \in \mathcal{M}, i \in \mathcal{K}, \forall c, t \\
& C_5: \sum_{m\in\mathcal{M}} \sum_{c\in\mathcal{C}_m} x_{kmc} \leq 1, \forall k, \\
& C_6: \mathcal{F}_k(t) \geq f'_k(t) \sum_{m\in\mathcal{M}} \sum_{c\in\mathcal{C}_m} x_{kmc}, \forall k \\
& C_7: x_{kmc} \in \{0, 1\}, \forall t, m, k, c.
\end{aligned} \tag{9}$$

Remark. The constraint reduction facilitates solving the problem in two ways: (1) It decouples the optimization of the two variables x_{kmc} and $\mathcal{F}_k$, enabling the competitive algorithm design which we present in Section III-B. (2) The approximated feasible region is simplified and can be sufficiently represented by multiple submodular constraints, which facilitates the analysis of the worst-case performance bound in Section III-E.

The proposed *EdgeOPT* algorithm schedules the online tasks by maintaining a solution within the feasible region of (9). It defines an adaptive threshold to determine whether and where to schedule each new request, and in which channel to transmit the data if the task is accepted. The computing resources are dynamically allocated by solving a novel convex programme with water-filling structure. It also integrates a subroutine for checking the feasibility of each candidate schedule.

B. Parallel Scheduling

The schedule is determined by the decision of task dispatching, the assignment of communication channel, and the temporal management of the computing resources.

1) Task Dispatching and Channel Assignment: We define a tuple $\boldsymbol{\varphi}_k := (m_k, c_k)$, where $m_k \in \mathcal{M}$ and $c_k \in \mathcal{C}$ specify the dispatched server and the assigned communication channel of task k, respectively. At each time t, the optimization for the scheduling decisions can be decoupled from (9), and is given by:

$$\max_{x_{kmc}\in\{0,1\}} \left\{ \sum_{k\in\mathcal{K}} \mathcal{R}_k \sum_{m\in\mathcal{M}} \sum_{c\in\mathcal{C}_m} x_{kmc} : C_1 \sim C_5 \text{ in (9)} \right\}, \tag{10}$$

which is essentially a variant of the online submodular knapsack problem [44], with items being enabled to transit between different states and each state only occupies a subset of the knapsack dimensions. Special cases for this problem are always handled with threshold based policies given some prior global information [45], [46], [47]. We propose an adaptive threshold based policy that deals with state transition and enables efficient resource allocations for all accepted tasks.

We use $\mathcal{J} = \{\mathcal{C}, \mathcal{F}, \mathcal{D}, \mathcal{K}_+\}$ to denote the set of constraints, whose elements represent the channel, computation, memory and the interference constraint at each active user, respectively. The function $u_j : 2^{\mathcal{K}} \mapsto \mathbb{R}, \forall j \in \mathcal{J}$ maps each set of user to their utilization of the type-j resource:

$$u_j(\mathcal{S}_t^m) = \sum_{k\in\mathcal{S}_t^m} q_j(\boldsymbol{\varphi}_k, t), \tag{11}$$

where $q_j(\boldsymbol{\varphi}_k, t)$ denotes the resource demand for task k in dimension j at t under decision $\boldsymbol{\varphi}_k$. For $j \in \mathcal{J} \setminus \mathcal{D}$: $q_{\mathcal{C}}(\boldsymbol{\varphi}_k, t) = \mathbb{1}_{t\in[a_k, a_k+\hat{\mathfrak{r}}_{mk}]}$, $q_{\mathcal{F}}(\boldsymbol{\varphi}_k, t) = f_k'(t)\mathbb{1}_{t\in[a_k+v_{mk}, d_k]}$, and $q_i(\boldsymbol{\varphi}_k, t) = |h_{m_ik}|^2 p_k \mathbb{1}_{c_k=c_i}\mathbb{1}_{t\in[a_k, a_k+\hat{\mathfrak{r}}_{mk}]}$, $\forall i \in \mathcal{K}_+$. For memory resource, the consumption is submodular. To express the consumption using the same expression as (11), we sort the users of $\mathcal{S}_t^m$ in the order they are accepted, and only the first users that request the containers of a new type has non-zero memory demand, *i.e.*, $q_{\mathcal{D}}(\boldsymbol{\varphi}_k, t) = \mathcal{D}_l \mathbb{1}_{o_{kl}^{mt}} \mathbb{1}_{t\in[a_k, d_k]}$, where o_{kl}^{mt} is the event that k is the first user in server m at time t to request a container of type l.

Denote by $\mathcal{E}_{t,m}^C$, $\mathcal{E}_{t,m}^F$, and $\mathcal{E}_{t,m}^Z$ the set of tasks that reach the specified transmission delays, finish execution, and complete container instantiation at time t, respectively. The server usage $u_j(\mathcal{S}_t^m), \forall j \in \mathcal{J}, m \in \mathcal{M}$, is updated upon each task state transition: whenever a new task is accepted, or at least one of $\mathcal{E}_{t,m}^B$, $\mathcal{E}_{t,m}^C$, $\mathcal{E}_{t,m}^Z$, and $\mathcal{E}_{t,m}^F$ is non-empty. At each time t a new request $k \in \mathcal{K}$ arrives, it can be scheduled to server $m \in \mathcal{M}$ using channel $c \in \mathcal{C}_m$ only if its reward $\mathcal{R}_k$ exceeds the corresponding threshold $\Phi_m(\mathcal{S}_t^m, \boldsymbol{\varphi}_k)$ and the decision satisfies all the constraints. The threshold is given by:

$$\Phi_m(\mathcal{S}_t^m, \boldsymbol{\varphi}_k) = \sum_{j\in\mathcal{J}} \gamma_j \hat{q}_j(\boldsymbol{\varphi}_k, t) \left(e^{\rho_j^m(\mathcal{S}_t^m)} - 1 \right), \tag{12}$$

with

$$\hat{q}_j(\boldsymbol{\varphi}_k, t) = q_j(\boldsymbol{\varphi}_k, t) + \epsilon' C_{j,t}^m \mathbb{1}_{q_j(\boldsymbol{\varphi}_k, t)=0}, \tag{13}$$

and

$$\rho_j^m(\mathcal{S}_t^m) = \left\lfloor \frac{u_j(\mathcal{S}_t^m)}{C_{j,t}^m} \ln(\eta\theta_t) \right\rfloor, \tag{14}$$

where $C_{j,t}^m$ is the capacity of type-j resource in server m at time t, ϵ' is a sufficiently small constant to reduce the threshold when the type-j resource can be shared among tasks, *i.e.*, when $q_j(\boldsymbol{\varphi}_k, t) = 0$. Note that the capacity is dependent of t since $\mathcal{J}$ also incorporates the interference constraints of the active users that may finish transmissions. Therefore $C_{j,t}^m = \hat{I}_k$ when $j \in \bigcup_{m\in\mathcal{M}} S_{t,m}^B$ and $C_{j,t}^m = \Lambda \to \infty$, otherwise. $\{\gamma_j\}_{\forall j\in\mathcal{J}}$ are parameters that balance the contribution of each dimension, η is a parameter that scales the dependence of $\rho_j^m(S_t^m)$ on $u_j(S_t^m)$. Define the resource intensity $\sigma_k(t)$ of task k as the ratio of its reward to the weighted sum of resource demand at time t, we have $\sigma_k(t) = \mathcal{R}_k / \sum_j \gamma_j q_j(\boldsymbol{\varphi}_k, t)$. We use θ_t to keep track of the maximum resource intensity fluctuation up to time t, which is calculated as:

$$\theta_t = \max_{k,l\in\mathcal{K}} \left\{ \frac{\sigma_k(a_k)}{\sigma_l(a_l)} : a_k, a_l \le t \right\}, \tag{15}$$

and is updated upon each task arrival to learn the heterogeneity of resource intensity. This helps approximate the global intensity fluctuation before all the tasks are revealed, allowing better control over the aggressiveness of the proposed threshold.

Intuitively, the threshold $\Phi^m(\mathcal{S}_t^m, \boldsymbol{\varphi}_k)$ scales with the server usage to prioritize high-demand tasks when the system is heavily loaded. The main scheduling procedure is outlined in *Algorithm 1*. Note that when a new task $k \in \mathcal{K}$ queries the system, the algorithm dispatches the task to server m^* and transmit its data in channel c^* that minimizes the scheduling cost: $\Phi^m(\mathcal{S}_t^m, \boldsymbol{\varphi}_k) + \mathcal{R}_k \mathbb{1}_{\mathcal{S}_t^m \cup k \notin \mathcal{I}_m}$, which equals to the threshold value when $\boldsymbol{\varphi}_k$ is feasible, and is no less than $\mathcal{R}_k$ if the decision is infeasible. The request can be scheduled only when its reward is no less than such quantity, indicating that the reward can cover the minimum cost the decision may incur by taking into consideration all the constraints as well as the potential cold-start latency at the dispatched server.

2) Computing Resource Allocation: The computing resources $\mathcal{F}_k$ are dynamically allocated at each time t by minimizing the sum remaining precessing delay, while guaranteeing that all tasks can be completed before their deadlines. The decoupled optimization problem is given by:

$$\min_{\mathcal{F}_k \ge f_k'(t)} \left\{ \sum_{k\in\bar{\mathcal{S}}_t^m \setminus \mathcal{S}_{m,t}^C} \frac{w_k'(t)}{\mathcal{F}_k} : \sum_{k\in\bar{\mathcal{S}}_t^m \setminus \mathcal{S}_{m,t}^C} \mathcal{F}_k \le \mathcal{F}^m \right\}, \tag{16}$$

where $\bar{\mathcal{S}}_t^m$ is the tasks that have finished container start-ups and $w_k'(t)$ is the *remaining* workload for task $k \in \bar{\mathcal{S}}_t^m$ at time t. The algorithm buffers the transmitted requests and only processes the tasks that have reached their specified transmission delays to combat the time varying interference uncertainties brought by the online user arrivals. Eq. (16) defines a convex optimization problem with water-filling structure [48], which can be solved efficiently using standard convex optimization techniques. The Lagrangian of (16) is given by:

$$\mathcal{L}(\boldsymbol{\mathcal{F}}, \boldsymbol{\lambda}, \mu) = \sum_{k\in\bar{\mathcal{S}}_t^m \setminus \mathcal{S}_{m,t}^C} \frac{w_k'(t)}{\mathcal{F}_k}$$

Algorithm 1 Online Parallel Task Scheduling

Input : Hyper-parameters $\eta, \gamma_j, \forall j \in \mathcal{J}$ and a set $\mathcal{K}$ of tasks arriving online
Output: Set $\mathcal{S}_t^m$ of accepted tasks

Initialization: $\mathcal{S}_t^m \leftarrow \varnothing$, $\bar{\mathcal{S}}_t^m \leftarrow \varnothing$, $\theta_t \leftarrow 1$
for every t **do**
  Update system information at each $m \in \mathcal{M}$
  if $\mathcal{E}_{t,m}^F \neq \varnothing$ **then**
    $\mathcal{S}_t^m \leftarrow \mathcal{S}_t^m \setminus \mathcal{E}_{t,m}^F$, $\bar{\mathcal{S}}_t^m \leftarrow \bar{\mathcal{S}}_t^m \setminus \mathcal{E}_{t,m}^F$
    $u_j(\mathcal{S}_t^m) \leftarrow \sum_{k \in \mathcal{S}_t^m} q_j(\boldsymbol{\varphi}_k, t), \forall j \in \mathcal{J} \setminus \{\mathcal{C}, \mathcal{K}_+\}$
    $\rho_j^m(\mathcal{S}_t^m) \leftarrow \left\lfloor \frac{u_j(\mathcal{S}_t^m)}{C_{j,t}^m} \ln(\eta\theta_t) \right\rfloor, \forall j \in \mathcal{J} \setminus \{\mathcal{C}, \mathcal{K}_+\}$
    Reallocate $\{\mathcal{F}_k^*\}$, $\forall k \in \bar{\mathcal{S}}_t^m \setminus \mathcal{S}_{m,t}^C$ via **Alg. 2**
  if $\mathcal{E}_{t,m}^Z \neq \varnothing$ **or** $\mathcal{E}_{t,m}^C \neq \varnothing$ **then**
    $\mathcal{S}_{t,m}^C \leftarrow \mathcal{S}_{t,m}^C \setminus \mathcal{E}_{t,m}^C$, $\bar{\mathcal{S}}_t^m \leftarrow \bar{\mathcal{S}}_t^m \cup \mathcal{E}_{t,m}^Z$
    $u_j(\mathcal{S}_t^m) \leftarrow \sum_{k \in \mathcal{S}_t^m} q_j(\boldsymbol{\varphi}_k, t), \forall j \in \mathcal{J}$
    $\rho_j^m(\mathcal{S}_t^m) \leftarrow \left\lfloor \frac{u_j(\mathcal{S}_t^m)}{C_{j,t}^m} \ln(\eta\theta_t) \right\rfloor, \forall j \in \mathcal{J}$
    Allocate $\{\mathcal{F}_k^*\}$, $\forall k \in \bar{\mathcal{S}}_t^m \setminus \mathcal{S}_{t,m}^C$ via **Alg. 2**
  foreach new request $\{i\} \in \mathcal{K}$ arrives at t **do**
    Update θ_t as defined in (15)
    $\Phi^* \leftarrow \min_{\boldsymbol{\varphi}_i} \left\{ \Phi^m(\mathcal{S}_t^m, \boldsymbol{\varphi}_i) + \mathcal{R}_i \mathbb{1}_{\mathcal{S}_t^m \cup k \notin \mathcal{I}_m} \right\}$
    $(m^*, c^*) \leftarrow$ The minimizer in *line 16*
    if $\mathcal{R}_i \geq \Phi^*$ **then**
      Dispatch to m^*, initialize container if needed $\mathcal{S}_t^{m^*} \leftarrow \mathcal{S}_t^{m^*} \cup \{i\}$
      Transmit data in c^*: $\mathcal{S}_{t,m^*}^B \leftarrow \mathcal{S}_{t,m^*}^B \cup \{i\}$
      $u_j(\mathcal{S}_t^m) \leftarrow \sum_{k \in \mathcal{S}_t^m} q_j(\boldsymbol{\varphi}_k, t), \forall j \in \{\mathcal{C}, \mathcal{D}\}$
      $\rho_j^m(\mathcal{S}_t^m) \leftarrow \left\lfloor \frac{u_j(\mathcal{S}_t^m)}{C_{j,t}^m} \ln(\eta\theta_t) \right\rfloor, \forall j \in \{\mathcal{C}, \mathcal{D}\}$
    else
      Reject this request

$$+ \sum_{k \in \bar{\mathcal{S}}_t^m \setminus \mathcal{S}_{m,t}^C} \lambda_k \left[f_k'(t) - \mathcal{F}_k \right] + \mu \left[\sum_{k \in \bar{\mathcal{S}}_t^m \setminus \mathcal{S}_{m,t}^C} \mathcal{F}_k - \mathcal{F}^m \right], \tag{17}$$

where λ_k and μ are the Lagrange multipliers. The optimizer $\mathcal{F}_k^*$ satisfies the Karush-Kuhn-Tucker conditions [49]:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathcal{F}_k} &= -\frac{w_k'(t)}{\mathcal{F}_k^2} - \lambda_k + \mu = 0, \forall k \in \bar{\mathcal{S}}_t^m \setminus \mathcal{S}_{m,t}^C, \\ \lambda_k &\left(f_k'(t) - \mathcal{F}_k \right) = 0, \forall k \in \bar{\mathcal{S}}_t^m \setminus \mathcal{S}_{m,t}^C, \\ \mu &\left(\sum_{k \in \bar{\mathcal{S}}_t^m \setminus \mathcal{S}_{m,t}^C} \mathcal{F}_k - \mathcal{F}^m \right) = 0, \ \boldsymbol{\lambda}, \mu \succeq 0, \end{aligned} \tag{18}$$

solving which yields:

$$\mathcal{F}_k^* = \max \left\{ \frac{\sqrt{w_k'(t)} \left(\mathcal{F}^m - \sum_{k \in \mathcal{X}} f_k'(t) \right)}{\sum_{k \in (\bar{\mathcal{S}}_t^m \setminus \mathcal{S}_{m,t}^C) \setminus \mathcal{X}} \sqrt{w_k'(t)}}, f_k'(t) \right\}, \tag{19}$$

where $\mathcal{X}$ satisfies $\frac{\sqrt{w_k'(t)}}{\sqrt{\mu}} < f_k'(t), \forall k \in \mathcal{X}$. The procedure for finding the set $\mathcal{X}$ and $\mathcal{F}_k^*$ is summarized in *Algorithm 2*.

Algorithm 2 Resource Allocation Subroutine at $m \in \mathcal{M}$

Input : $\bar{\mathcal{S}}_t^m$, $\mathcal{S}_{m,t}^C$, $\mathcal{F}^m$ and computation demand $\mathbf{w}'(t), \mathbf{f}'(t)$
Output: Optimal allocation $\{\mathcal{F}_k^*\}$

$\mathcal{S} \leftarrow \bar{\mathcal{S}}_t^m \setminus \mathcal{S}_{m,t}^C$
$\mathcal{X} \leftarrow \varnothing$
$\frac{1}{\sqrt{\mu}} \leftarrow \frac{\mathcal{F}^m - \sum_{k \in \mathcal{X}} f_k'(t)}{\sum_{k \in \mathcal{S} \setminus \mathcal{X}} \sqrt{w_k'(t)}}$
$\hat{k} \leftarrow \arg\max_{k \in \mathcal{S} \setminus \mathcal{X}} \{ \frac{f_k'(t)}{\sqrt{w_k'(t)}} \}$
while $\frac{f_{\hat{k}}'(t)}{\sqrt{w_{\hat{k}}'(t)}} \geq \frac{1}{\sqrt{\mu}}$ **do**
  $\mathcal{X} \leftarrow \mathcal{X} \cup \hat{k}$
  $\frac{1}{\sqrt{\mu}} \leftarrow \frac{\mathcal{F}^m - \sum_{k \in \mathcal{X}} f_k'(t)}{\sum_{k \in \mathcal{S} \setminus \mathcal{X}} \sqrt{w_k'(t)}}$
  $\hat{k} \leftarrow \arg\max_{k \in \mathcal{S} \setminus \mathcal{X}} \{ \frac{f_k'(t)}{\sqrt{w_k'(t)}} \}$
return $\mathcal{F}_k^* = \max\{ \frac{\sqrt{w_k'(t)}}{\sqrt{\mu}}, f_k'(t) \}, \forall k \in \mathcal{S}$

Algorithm 3 Feasibility Oracle for $i \in \mathcal{K}$ and $m \in \mathcal{M}$

Input : Task information in $\mathcal{S}_t^m \cup \{i\}$
Output: Whether or not $\mathcal{S}_t^m \cup \{i\} \in \mathcal{I}_m$

$\mathcal{S}_t^m \leftarrow \mathcal{S}_t^m \cup \{i\}$
while $t \in [a_i, a_i + \tau_i]$ **do**
  Go to the next t when any of $\mathcal{E}_{t,m}^F$, $\mathcal{E}_{t,m}^Z$ and $\mathcal{E}_{t,m}^C$ is non-empty under current allocation $\{\mathcal{F}_k^*\}, \forall k \in \mathcal{S}_t^m$
  Update $\mathcal{S}_t^m$, $\bar{\mathcal{S}}_t^m$, and $\mathcal{S}_{t,m}^C$ based on the definitions
  Update $f_k'(t), \forall k \in \mathcal{S}_t^m$ according to (6)
  Update $\{\mathcal{F}_k^*\}, \forall k \in \bar{\mathcal{S}}_t^m \setminus \mathcal{S}_{m,t}^C$ via **Algorithm 2**
  if $\exists j \in \mathcal{J} \setminus \{\mathcal{C}, \mathcal{K}_+\}$ *that is violated* **then**
    return FALSE
return TRUE

C. Feasibility Check

Denote by $\mathcal{I}_m$ the set of feasible decisions of server m. The feasibility subroutine described in *Algorithm 3* firstly assumes the acceptance of a new task, it then checks the feasibility using the updated resource allocation iteratively during all the task states from its arrival to departure. It guarantees that a newly scheduled task will neither lead to the expiration of accepted tasks due to limited computing resource, nor result in capacity overflow of both memory and computing resource.

D. Complexity Analysis

Theorem 1: The time complexity of EdgeOPT for scheduling each task is $\mathcal{O}\left(|\mathcal{M}|\left(|\mathcal{S}_t^m|^2 + |\mathcal{C}|\right)\right)$.

Proof: To obtain the scheduling decision for each new request, the algorithm spends $|\mathcal{M}|$ iterations computing the cost at each server. Within each iteration, the feasibility oracle iterates at most $3|\mathcal{S}_t^m| + 3$ times to verify the feasibilities during all task states. Resource allocation can be completed with at most $|\mathcal{S}_t^m|$ iterations. Moreover, $|\mathcal{C}|$ iterations are

also needed to search for the best combination of server and communication channel. This leads to a time complexity of $\mathcal{O}\left(|\mathcal{M}|\left(|\mathcal{S}_t^m|^2+|\mathcal{C}|\right)\right)$ for *EdgeOPT* to schedule each task. ■

Note that $|\mathcal{S}_t^m|$ is bounded in practice due to the limited edge capacity, and if we further restrict the candidate site of servers to be the nearest server, *EdgeOPT* can be more computationally efficient.

E. Competitive Ratio Analysis

We prove the competitive ratio of *EdgeOPT* by investigating the ratio between the upper bound of the offline optimum and the lower bound of the algorithm's output within each carefully partitioned time intervals. The main theorem is as follows.

Theorem 2: Assume there exists some types of resources such that the consumption ratio for any task $i \in \mathcal{K}$ is bounded: $\epsilon' \leq \hat{q}_j(\boldsymbol{\varphi}_i, t)/C_j^m \leq \epsilon$, for some $j \in \mathcal{J}$. If we choose η such that $\epsilon < \min\left\{\frac{1}{2}-\frac{1}{2\ln(\eta)}, \frac{1}{\ln(\eta\theta)}\right\}$, the competitive ratio of EdgeOPT is $\mathcal{O}\left(\lambda\theta\ln(\theta)\right)$, where $\lambda = \max_{k,l}\left(\frac{\tau_k}{\tau_l}\right)$ is the deadline fluctuation, $\theta = \max_t \theta_t$.

The following lemma allows the competitive ratio of multiple servers to be expressed in terms of a single server with the largest competitive ratio, in which case the proof can be greatly simplified.

Lemma 1: Let $c_{\mathcal{N}}$ be the competitive ratio of Algorithm 1 when only servers in set $\mathcal{N} \subseteq \mathcal{M}$ exist. It follows that:

$$c_{\mathcal{M}} \leq 1 + \max_{m\in\mathcal{M}} c_m. \tag{20}$$

Proof: The lemma can be proved by treating each server as a knapsack and applying the same procedure as *Lemma 2* in [46]. ■

The single server case can be analyzed by exploiting the behavior of $\rho_j^m(S_t^m)$ using a novel time partition technique. Specifically, time is partitioned into consecutive slots in which $\forall j \in \mathcal{J}, \rho_j^m(S_t^m)$ stays constant. Define *receiving slot* as the slot where at least one task is accepted. The slots are grouped into intervals, each starts with a *receiving slot*, and ends when any of the following happens:

- The present slot is not *receiving slot* whereas the next slot is *receiving slot*.
- Both the present slot and the next slot are *receiving slot*, and $\exists j \in \mathcal{J}$ such that $\rho_j^m(S_t^m)$ decreases in the next slot.
- Both the present slot and the next slot are *receiving slot*, and the length of the interval reaches $\tau_{\max}$.
- Some of the accepted requests finishes transmission.

The partition can be illustrated by Fig. 2. Denote by $\mathcal{T}$ the set of intervals resulted from such partition. We have two immediate observations. Firstly, as no request finishes transmission within each interval, the capacity remains constant for every j. We therefore neglect the subscript t in $C_{j,t}^m$ and use C_j^m instead when there is no confusions. Secondly, every interval contains consecutive *receiving slots* where $\rho_j(S_t)$ is non-decreasing, which may be followed by consecutive non-receiving slots where $\rho_j^m(S_t^m)$ is non-increasing. The duration of each interval is bounded by:

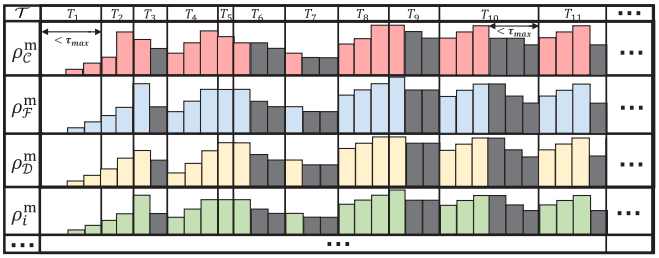

Fig. 2. The evolution of $\rho_j^m(S_t^m)$ through different interval. The colored slots are receiving slots and the shaded ones are non-receiving slots.

Lemma 2: The duration of each interval is upper-bounded by $2\tau_{max}$, where τ_{max} is the maximum allowable task delay.

Proof: See Appendix A for the proof. ■

Denote by $\hat{O}_T^m$ the optimal set and $\tilde{S}_T^m$ the set accepted by the algorithm at the end of interval $T \in \mathcal{T}$ for server $m \in \mathcal{M}$. When there is no confusions, we define $q_{ij}^m := \hat{q}_j(\boldsymbol{\varphi}_i, a_i)$.

Lemma 3: For $\forall T \in \mathcal{T}, j \in \mathcal{J}, m \in \mathcal{M}$, it holds that:

$$\sum_{i\in\hat{O}_T^m} q_{ij}^m \leq \left(\frac{1}{\epsilon'}+\frac{2\lambda}{\epsilon'^2}\right) C_j^m. \tag{21}$$

Proof: See Appendix B for the proof. ■

Let $f(\cdot)$ be the objective function. For each $m \in \mathcal{M}$, denote by $S'_{m,T}$ the set of tasks being served at the end of interval T, and $S_{T,i}^m$ the set of currently serving tasks after the i-th task is accepted during interval T, we have $S_{T,0}^m = S'_{m,T-1}$. Define $S_{m,T}^* = S_{T,|\tilde{S}_T^m|}^m$, which maximizes $\rho_j^m(S_t^m), \forall j \in \mathcal{J}$ for $t \in T$. The competitive ratio can be derived by lower bounding the objective $f(\tilde{S}_T)$ and upper bounding $f(\hat{O}_T)$ for each $T \in \mathcal{T}$. The competitive ratio when there only exists server m can be proved with the help of the following proposition:

Proposition 1: If there only exists server $m \in \mathcal{M}$, then for $\forall T \in \mathcal{T}$, we have $\frac{f(\hat{O}_T)}{f(\tilde{S}_T^m)} \leq \mathcal{O}\left(\lambda\theta^{[m]}\ln(\theta^{[m]})\right)$, where $\theta^{[m]} = \max_{t\in T}\theta_t$ for all tasks accepted at server m.

The proposition is proved under two cases.

Case 1: $C_j^m - u_j(S_t^m) > \epsilon C_j^m$ for all $j \in \mathcal{J}$ and $t \in T$.

In this case, we have $\rho_j^m(S_{m,T}^*) \leq \lfloor\frac{u_j(S_{m,T}^*)}{C_j^m}\ln(\theta^{[m]}\eta)\rfloor < \lfloor(1-\epsilon)\ln(\theta^{[m]}\eta)\rfloor < \ln(\theta^{[m]}\eta), \forall j \in \mathcal{J}$. We first consider tasks in $\hat{O}_T^m \setminus \tilde{S}_T^m$. The following lemma bounds $f(\hat{O}_T^m \setminus \tilde{S}_T^m)$:

Lemma 4: In Case 1, we have:

$$f(\hat{O}_T^m \setminus \tilde{S}_T^m) \leq \left(\frac{1}{\epsilon'}+\frac{2\lambda}{\epsilon'^2}\right)\sum_{j\in\mathcal{J}}\gamma_j C_j^m\left(e^{\rho_j^m(S_{m,T}^*)}-1\right). \tag{22}$$

Proof: See Appendix C for the proof. ■

Lemma 5: In Case 1, define $\Delta_{T,j}^m(S) = \rho_j^m(S) - \rho_j^m(S'_{m,T-1})$ and $\mathcal{J}_n^T = \left\{j \in \mathcal{J} : \Delta_{T,j}^m(S_{m,T}^*) \geq n\right\}$. The value of $f(\tilde{S}_T^m)$ can be lower bounded by:

$$f(\tilde{S}_T^m) \geq \sum_{j\in\mathcal{J}\setminus\mathcal{J}_1^T}(e^{\rho_j^m(S'_{m,T-1})}-1)\epsilon'\gamma_j C_j^m + [e-1]\sum_{j\in\mathcal{J}_1^T\setminus\mathcal{J}_2^T}\gamma_j\epsilon' C_j^m + \min\left\{\mathcal{L}'_{T,2}, \mathcal{L}_{T,2}\right\}, \tag{23}$$

where

$$\mathcal{L}'_{T,2} = [e-1] \sum_{j\in\mathcal{J}_2^T} \epsilon' \gamma_j C_j^m,$$

$$\mathcal{L}_{T,2} = \frac{\left[e^2-2(e-1)-1\right]}{e-1} \sum_{j\in\mathcal{J}_2^T} \gamma_j C_j^m \left(\frac{1}{\ln(\theta^{[m]}\eta)} - \epsilon\right).$$

Proof: Let θ_i^m be the maximum resource intensity up to the acceptance of task i at server m. Since the reward of any accepted task must exceed the threshold, we have:

$$f(\tilde{S}_T^m) = \sum_{i\in\tilde{S}_T^m} \mathcal{R}_i \geq \sum_{i\in\tilde{S}_T^m}\sum_{j\in\mathcal{J}} \gamma_j q_{ij}^m \left(e^{\rho_j^m(S_{T,i-1}^m)} - 1\right)$$
$$= \sum_{i\in\tilde{S}_T^m}\sum_{j\in\mathcal{J}} \gamma_j q_{ij}^m \left(e^{\left\lfloor \frac{u_j(S_{T,i-1}^m)}{C_j^m}\ln(\theta_i^m\eta)\right\rfloor} - 1\right). \quad (24)$$

Note that the function $h(u(S_{T,i}^m)) = \left\lfloor \frac{u_j(S_{T,i}^m)}{C_j^m}\ln(\theta_i^m\eta)\right\rfloor$ is piecewise constant, and the step length of the function is $C_j^m/\ln(\theta_i^m\eta)$, which is non-increasing since θ_i^m is non-decreasing during execution. Denote by $S_{T,n}^{m,j} \subseteq \tilde{S}_T^m$ the largest set that satisfies $\Delta_{T,j}^m(S_{T,n}^{m,j}) \leq n$. This gives:

$$\begin{aligned}
&f(\tilde{S}_T^m)\\
&\geq \sum_{i\in\tilde{S}_T^m}\sum_{j\in\mathcal{J}} \gamma_j q_{ij}^m \left(e^{\left\lfloor \frac{u_j(S_{T,i-1}^m)}{C_j^m}\ln(\theta_i^m\eta)\right\rfloor} - 1\right)\\
&= \sum_{j\in\mathcal{J}} \sum_{n=\rho_j^m(S'_{m,T-1})}^{\rho_j^m(S_{m,T}^*)} (e^n-1) \left(\sum_{i\in Z_{T,n}^{m,j}\setminus Z_{T,n-1}^{m,j}} \gamma_j q_{ij}^m\right)\\
&\overset{\text{def}}{=} \mathcal{L}'_{T,1} + \sum_{j\in\mathcal{J}_2^T} \sum_{n=\rho_j^m(S'_{m,T-1})}^{\rho_j^m(S_{m,T}^*)} (e^n-1) \sum_{i\in Z_{T,n}^{m,j}\setminus Z_{T,n-1}^{m,j}} \gamma_j q_{ij}^m\\
&\geq \mathcal{L}'_{T,1} + \sum_{j\in\mathcal{J}_2^T} \sum_{n=0}^{\Delta_{T,j}^m(S_{m,T}^*)} (e^n-1) \left(\sum_{i\in S_{T,n}^{m,j}\setminus S_{T,n-1}^{m,j}} \gamma_j q_{ij}^m\right)\\
&\overset{\text{def}}{=} \mathcal{L}'_{T,1} + \mathcal{L}(\mathcal{J}_2^T), \quad (25)
\end{aligned}$$

where $Z_{T,n}^{m,j} \overset{\text{def}}{=} S_{T,n-\rho_j(S'_{T-1})}^{m,j}$ and $\mathcal{L}(\cdot)$ is a set function defined in the last line of (25). The lower bound $\mathcal{L}_{T,1}$ for $\mathcal{L}'_{T,1}$ can be derived as:

$$\begin{aligned}
\mathcal{L}'_{T,1} =& \sum_{j\in\mathcal{J}\setminus\mathcal{J}_1^T} \gamma_j \left[e^{\rho_j^m(S'_{m,T-1})} - 1\right] \sum_{i\in S_{T,0}^{m,j}} q_{ij}^m\\
&+ \left[e^{\rho_j^m(S_{m,T}^*)} - 1\right] \sum_{j\in\mathcal{J}_1^T\setminus\mathcal{J}_2^T} \gamma_j \sum_{i\in S_{T,1}^{m,j}\setminus S_{T,0}^{m,j}} q_{ij}^m\\
\geq& \sum_{j\in\mathcal{J}\setminus\mathcal{J}_1^T} (e^{\rho_j^m(S'_{m,T-1})} - 1)\epsilon'\gamma_j C_j^m\\
&+ [e-1] \sum_{j\in\mathcal{J}_1^T\setminus\mathcal{J}_2^T} \epsilon'\gamma_j C_j^m, \quad (26)
\end{aligned}$$

where the inequality is because: (i) for all $j \in \mathcal{J}_1^T \setminus \mathcal{J}_2^T$, $\rho_j^m(S_{m,T}^*) \geq 1$, (ii) for all $j \in \mathcal{J} \setminus \mathcal{J}_2^T$ the aggregate utilization ratio $\sum_{i\in S_{T,1}^{m,j}\setminus S_{T,0}^{m,j}} \frac{q_{ij}^m}{C_j^m}$ and $\sum_{i\in S_{T,0}^{m,j}} \frac{q_{ij}^m}{C_j^m}$ are lower bounded by the minimum single task utilization ratio $\epsilon' = \inf_{i\in\mathcal{K},j\in\mathcal{J},m\in\mathcal{M}} \left[\frac{q_{ij}^m}{C_j^m}\right]$.

Next, we lower bound $\mathcal{L}(\mathcal{J}_2^T)$. For $j \in \mathcal{J}_2^T$, $S_{T,n}^{m,j} \setminus S_{T,n-1}^{m,j}$ can be empty for some $n \in \mathbb{N}$, the lower bound can be derived by focusing on the boundaries where $\left\lfloor \frac{u_j(S_{T,i}^m)}{C_j^m}\ln(\theta_i^m\eta)\right\rfloor$ increases, which includes the following cases:

(i) $S_{T,n}^{m,j} \setminus S_{T,n-1}^{m,j} \neq \varnothing$ for every $0 \leq n \leq \Delta_{T,j}^m(S_{m,T}^*)$:

In this case, $\rho_j^m(S_{T,|S_{T,n}^j|}^m) - \rho_j(S_{T,|S_{T,n-1}^j|}^m) = 1$ for all $0 \leq n \leq \Delta_{T,j}^m(S_{m,T}^*)$. Denote the first and the last task in $S_{T,n}^{m,j}$ as $e_T^j(n)$ and $l_T^j(n)$, respectively. For $u(S_t^m) \in [u(S_{T,|S_{T,n}^j|}^m), u(S_{T,|S_{T,n-1}^j|}^m)]$, the step-length of $h(u(S_t^m))$ is lower bounded by $\frac{C_j^m}{\ln(\theta_{l_T^j(n)}^m\eta)}$ since θ_i^m is non-decreasing. Moreover, the step-length is no larger than $q_j(\boldsymbol{\varphi}_{e_T^j(n)}, a_{e_T^j(n)}) + \sum_{i\in S_{T,n}^{m,j}\setminus S_{T,n-1}^{m,j}} q_j(\boldsymbol{\varphi}_i, a_i)$, since otherwise we have $e_T^j(n) \leq l_T^j(n-1)$, which contradicts the definition of $S_{T,n}^{m,j}$, $e_T^j(n)$ and $l_T^j(n)$. It implies:

$$q_j(\boldsymbol{\varphi}_{e_T^j(n)}, a_{e_T^j(n)}) + \sum_{i\in S_{T,n}^{m,j}\setminus S_{T,n-1}^{m,j}} q_j(\boldsymbol{\varphi}_i, a_i) \geq \frac{C_j}{\ln(\theta_{l_T^j(n)}\eta)}. \quad (27)$$

Therefore, when $S_{T,n}^{m,j} \subsetneq \tilde{S}_T^m$, we have:

$$\begin{aligned}
\sum_{i\in S_{T,n}^{m,j}\setminus S_{T,n-1}^{m,j}} q_j(\boldsymbol{\varphi}_i, a_i) &\geq \frac{C_j^m}{\ln(\theta_{l_T^j(n)}^m\eta)} - q_j(\boldsymbol{\varphi}_{e_T^j(n)}, a_{e_T^j(n)})\\
&\geq \frac{C_j^m}{\ln(\theta^{[m]}\eta)} - q_j(\boldsymbol{\varphi}_{e_T^j(n)}, a_{e_T^j(n)})\\
&\geq \left(\frac{1}{\ln(\theta^{[m]}\eta)} - \epsilon\right) C_j^m, \quad (28)
\end{aligned}$$

where the second inequality is from the definition of $\theta^{[m]}$, the last inequality is because each $q_j(\boldsymbol{\varphi}_i, a_i)$ is upper bounded by ϵC_j^m.

(ii) $S_{T,n}^{m,j} \setminus S_{T,n-1}^{m,j} = \varnothing$ for some $0 \leq n \leq \Delta_{T,j}^m(S_{m,T}^*)$:

In this case, we have $S_{T,n}^{m,j} = S_{T,n-1}^{m,j}$ for some $0 \leq n \leq \Delta_{T,j}^m(S_{m,T})$. Suppose there are $H_T^{m,j}$ distinct $S_{T,n}^{m,j}, n \in \mathbb{N}$, denoted each of them in an inclusion-wise increasing order by $G_{T,h}^{m,j}, h = \{0,1,\ldots,H_T^{m,j}-1\}$. It is clear that $\Delta_{T,j}^m(S_{m,T}^*) \geq 2$ for $j \in \mathcal{J}_2^T$, we have $2 \leq H_T^{m,j} \leq \Delta_{T,j}^m(S_{m,T}^*) + 1$. Moreover, $\forall h < h'$, we have $G_{T,h}^{m,j} \subsetneq G_{T,h'}^{m,j}$ and $\Delta_{T,j}^m(G_{T,h}^{m,j}) < \Delta_{T,j}^m(G_{T,h'}^{m,j})$.

A trivial case is when $H_T^{m,j} = 2$, where there are only 2 different set $G_{T,0}^{m,j}$ and $G_{T,1}^{m,j}$. We get to the lower bound by taking $G_{T,0}^{m,j}$ and $G_{T,1}^{m,j}$ into $\mathcal{L}(\mathcal{J}_2^T)$:

$$\mathcal{L}(\mathcal{J}_2^T) \geq (e-1) \sum_{j\in\mathcal{J}_2^T} \gamma_j^m \epsilon' C_j^m \overset{\text{def}}{=} \mathcal{L}'_{T,2}. \quad (29)$$

It is worth noting that the competitive ratio can be generalized from the case when $H_T^{m,j} \geq 3$, thus the following proof only considers $H_T^{m,j} \geq 3$. When $H_T^{m,j} \geq 3$ and $G_{T,h}^{m,j} \subsetneq \tilde{S}_T^m$, we have the result similar to *Case (i)*:

$$\sum_{i\in G_{T,h}^{m,j}\setminus G_{T,h-1}^{m,j}} q_j(\boldsymbol{\varphi}_i, a_i) \geq \left(\frac{1}{\ln(\theta^{[m]}\eta)} - \epsilon\right) C_j^m. \quad (30)$$

Combining the results gives:

$$\begin{aligned}\mathcal{L}(\mathcal{J}_2^T) &= \sum_{j\in\mathcal{J}_2^T}\gamma_j^m \sum_{n=0}^{\Delta_{T,j}^m(S_{m,T}^*)}(e^n-1)\left(\sum_{i\in S_{T,n}^{m,j}\setminus S_{T,n-1}^{m,j}} q_{ij}^m\right)\\ &\geq \sum_{j\in\mathcal{J}_2^T}\gamma_j^m \sum_{h=0}^{H_T^{m,j}-1}(e^h-1)\left(\sum_{i\in G_{T,h}^{m,j}\setminus G_{T,h-1}^{m,j}} q_{ij}^m\right)\\ &\geq \sum_{j\in\mathcal{J}_2^T}\gamma_j^m\left(\frac{1}{\ln(\theta^{[m]}\eta)}-\epsilon\right)C_j^m\sum_{h=0}^{H_T^{m,j}-2}(e^h-1),\end{aligned} \tag{31}$$

where the first inequality is because the sum term equals to 0 when $S_{T,n}^{m,j}\setminus S_{T,n-1}^{m,j}=\varnothing$, and n must increase more than 1 to make such set non-empty. The last inequality is due to (28) and the fact that $q_{ij}^m \geq q_j(\boldsymbol{\varphi}_i,a_i)$. When $H_T^{m,j}\geq 3$, $\sum_{h=0}^{H_T^{m,j}-2}(e^h-1)=\frac{1}{e-1}\left(e^{H_T^{m,j}-1}-(e-1)(H_T^{m,j}-1)-1\right)$ is an increasing function of $H_T^{m,j}$. Since $H_T^{m,j}-1\geq 2$, it follows that:

$$\mathcal{L}(\mathcal{J}_2^T)\geq\frac{\left[e^2-2(e-1)-1\right]}{e-1}\sum_{j\in\mathcal{J}_2^T}\gamma_j^m C_j^m\left[\frac{1}{\ln(\theta^{[m]}\eta)}-\epsilon\right]\stackrel{\text{def}}{=}\mathcal{L}_{T,2}. \tag{32}$$

Summing up $\mathcal{L}_{T,1}$ and $\min\{\mathcal{L}'_{T,2},\mathcal{L}_{T,2}\}$ yields the lemma. ■

Fact 1: *For positive real numbers $a_1,a_2,\ldots,a_{|I|}$ and $b_1,b_2,\ldots,b_{|I|}$, it holds that:*

$$\frac{\sum_{i\in I}a_i}{\sum_{i\in I}b_i}\leq\max_{i\in I}\left\{\frac{a_i}{b_i}\right\}. \tag{33}$$

Proof of Proposition 1 under Case 1:

$$\begin{aligned}\frac{f(\hat{O}_T^m)}{f(\tilde{S}_T^m)} &= \frac{f(\tilde{S}_T^m)+f(\hat{O}_T^m\setminus\tilde{S}_T^m)}{f(\tilde{S}_T^m)}\\ &\leq 1+\frac{\left(\frac{1}{\epsilon'}+\frac{2\lambda}{\epsilon'^2}\right)\sum_{j\in\mathcal{J}}\gamma_j C_j^m\left(e^{\rho_j^m(S_{m,T}^*)}-1\right)}{\mathcal{L}_{T,1}+\mathcal{L}_{T,2}}\\ &= 1+\frac{\left(\frac{1}{\epsilon'}+\frac{2\lambda}{\epsilon'^2}\right)\sum_{\mathcal{J}\setminus\mathcal{J}_1^T}\gamma_j C_j^m(e^{\rho_j^m(S'_{m,T-1})}-1)}{\mathcal{L}_{T,1}+\mathcal{L}_{T,2}}\\ &\quad+\frac{\left(\frac{1}{\epsilon'}+\frac{2\lambda}{\epsilon'^2}\right)\sum_{\mathcal{J}_1^T\setminus\mathcal{J}_2^T}\gamma_j C_j^m(e^{\rho_j^m(S_{m,T}^*)}-1)}{\mathcal{L}_{T,1}+\mathcal{L}_{T,2}}\\ &\quad+\frac{\left(\frac{1}{\epsilon'}+\frac{2\lambda}{\epsilon'^2}\right)\sum_{j\in\mathcal{J}_2^T}\gamma_j C_j^m\left(e^{\rho_j^m(S_{m,T}^*)}-1\right)}{\mathcal{L}_{T,1}+\mathcal{L}_{T,2}},\end{aligned} \tag{34}$$

where $\mathcal{L}_{T,1}$ and $\mathcal{L}_{T,2}$ are taken from (26) and (32), respectively. Using *Fact 1*, the ratio $f(\hat{O}_T^m)/f(\tilde{S}_T^m)$ can be bounded as:

$$\begin{aligned}\frac{f(\hat{O}_T^m)}{f(\tilde{S}_T^m)}\leq 1+\left[\frac{1}{\epsilon'}+\frac{2\lambda}{\epsilon'^2}\right]\max\left\{\max_{j\in\mathcal{J}_1^T\setminus\mathcal{J}_2^T}\frac{e^{\rho_j^m(S_{m,T}^*)}-1}{\epsilon'(e-1)},\right.\\ \left.\max_{j\in\mathcal{J}_2^T}\left[\frac{(e-1)(e^{\rho_j^m(S_{m,T}^*)}-1)}{e^2-2(e-1)-1}\frac{\ln(\theta^{[m]}\eta)}{1-\epsilon\ln(\theta^{[m]}\eta)}\right],\frac{1}{\epsilon'}\right\}.\end{aligned} \tag{35}$$

Firstly, since $\rho_j^m(S_{m,T}^*)<\ln(\theta^{[m]}\eta),\forall j\in\mathcal{J}$ under *Case 1*, we have $e^{\rho_j^m(S_{m,T}^*)}-1\leq\theta^{[m]}\eta-1<\theta^{[m]}\eta$. Secondly, by assumption, ϵ is bounded such that $\epsilon<\frac{1}{\ln(\theta^{[m]}\eta)}$, therefore the quantity $\delta\stackrel{\text{def}}{=}\epsilon\ln(\theta^{[m]}\eta)<1$. Combining the results yields:

$$\begin{aligned}\frac{f(\hat{O}_T^m)}{f(\tilde{S}_T^m)}\leq 1+\left(\frac{1}{\epsilon'}+\frac{2\lambda}{\epsilon'^2}\right)\max\left\{\frac{1}{\epsilon'},\frac{\theta^{[m]}\eta}{\epsilon'(e-1)},\right.\\ \left.\frac{(e-1)\theta^{[m]}\eta}{e^2-2(e-1)-1}\frac{\ln(\theta^{[m]}\eta)}{1-\delta}\right\}=\mathcal{O}(\lambda\theta^{[m]}\ln(\theta^{[m]})).\end{aligned} \tag{36}$$

which completes the proof of *Case 1*. ■

Case 2: $\exists t\in T$ and $\mathcal{J}'_T$, $C_j^m-u_j(S_t^m)\leq\epsilon C_j^m,\forall j\in\mathcal{J}'_T$.

In this case, *Proposition 1* can be proved using the same ideas as *Case 1*, where we first find an upper bound for $f(\hat{O}_T^m)$ and a lower bound for $f(\tilde{S}_T^m)$, then leverage *Fact 1* to yield the competitive ratio.

Lemma 6: *The offline optimum in Case 2 is upper bounded as*:

$$f(\hat{O}_T^m)\leq\sigma'\left(\frac{1}{\epsilon'}+\frac{2\lambda}{\epsilon'^2}\right)\sum_{j\in\mathcal{J}_1^T\cup\mathcal{J}'_T}\gamma_j C_j^m, \tag{37}$$

where $\sigma'=\max_{i\in\mathcal{K}}\left(\mathcal{R}_i/\sum_{j\in\mathcal{J}_1^T\cup\mathcal{J}'_T}\gamma_j q_j(\boldsymbol{\varphi}_i,a_i)\right)$ is the maximum resource intensity over dimension $j\in\mathcal{J}_1^T\cup\mathcal{J}'_T$.

Proof: See Appendix D for the proof. ■

We leverage *lemma 6* to prove the bound of *Proposition 1* under *Case 2* using techniques similar to *Case 1*. For a detailed proof, we direct readers to Appendix E. Now we can obtain the competitive ratio by proving *Theorem 2*.

Proof of Theorem 2: Let $\hat{O}^{\mathcal{N}}$ be the optimal solution and $\tilde{S}^{\mathcal{N}}$ be the set of tasks accepted by servers in $\mathcal{N}\subseteq\mathcal{M}$ during the entire time horizon. By *Lemma 1*, we have:

$$\begin{aligned}\frac{f(\hat{O}^{\mathcal{N}})}{f(\tilde{S}^{\mathcal{N}})}&\leq 1+\max_{m\in\mathcal{M}}\frac{f(\hat{O}^m)}{f(\tilde{S}^m)}=1+\max_{m\in\mathcal{M}}\frac{f(\bigcup_{T\in\mathcal{T}}\hat{O}_T^m)}{f(\bigcup_{T\in\mathcal{T}}\tilde{S}_T^m)}\\ &=1+\max_{m\in\mathcal{M}}\frac{\sum_{T\in\mathcal{T}}f(\hat{O}_T^m)}{\sum_{T\in\mathcal{T}}f(\tilde{S}_T^m)}\\ &\leq 1+\max_{T\in\mathcal{T}}\left\{\max_{m\in\mathcal{M}}\mathcal{O}(\lambda\theta^{[m]}\ln(\theta^{[m]}))\right\}\\ &=\mathcal{O}(\lambda\theta\ln(\theta)),\end{aligned} \tag{38}$$

where the first inequality follows from *Lemma 1*, the second inequality follows from *Fact 1* and *Proposition 1*. ■

F. Scheduling for Tasks Composed of Multiple Functions

In this subsection, we discuss how to extend *EdgeOPT* to schedule virtual function chains. To this end, we define $\mathbf{L}_k=\{l_{k1},\ldots,l_{kL_k}\}$ to specify the function types requested for the service of user k, and let $\mathbf{W}_k=\{w_{k1},\ldots,w_{kL_k}\}$, $\boldsymbol{\tau}_k=\{\tau_{k1},\ldots,\tau_{kL_k}\}$ be the corresponding workloads and deadlines for each sub-function, respectively. The j-th function cannot be started before the $j-1$-th function is finished. Tasks composed of multiple function can therefore be represented

Algorithm 4 *EdgeOPT* for Function Chain Scheduling

Input : Hyper-parameters $\eta, \gamma_j, \forall j \in \mathcal{J}$ and a set $\mathcal{K}$ of tasks arriving online
Output: Set $\mathcal{S}_t^m$ of accepted tasks

Initialization: $\mathcal{S}_t^m \leftarrow \varnothing$, $\theta_t \leftarrow 1$
for every t **do**
  foreach new user $k \in \mathcal{K}$ at time t **do**
    Update θ_t as defined in (15)
    for $i = 1, \ldots, L_k$ **do**
      $\Phi^* \leftarrow \min_{\boldsymbol{\varphi}_{ki}} \Phi^m(\mathcal{S}_t^m, \boldsymbol{\varphi}_{ki}) + \mathcal{R}_i \mathbb{1}_{\mathcal{S}_t^m \cup k \notin \mathcal{I}_m}$
      if $i == 1$ **then**
        $(m_{k1}^*, c^*) \leftarrow$The minimizer in *line 6*
      else
        $m_{ki}^* \leftarrow$The minimizer in *line 6*
      if $\Phi^* > \mathcal{R}_k$ **then**
        $\mathcal{S}_t^m \leftarrow \mathcal{S}_t^m \setminus k, \forall m \in \mathcal{M}$
        Reject k and break the loop
      else
        $\mathcal{S}_t^{m_{ki}^*} \leftarrow \mathcal{S}_t^{m^*} \cup k$
  foreach task u that reaches the i-th deadline τ_{ui} **do**
    Forward the next sub-function to server $m_{u,i+1}^*$
  foreach $m \in \mathcal{M}$ **do**
    Update and reallocate resource via **Alg. 1**.

as $(s_k, \mathbf{W}_k, \mathbf{L}_k, \boldsymbol{\tau}_k, \mathcal{R}_k)$. Once task k is accepted, the system maps each sub-function to a server for execution, then the user transmits the input data sized s_k in the selected channel to the server assigned to the first virtual function, and the subsequent data between the first and the last sub-function are delivered through high-speed links with negligible delays. Reward $\mathcal{R}_k$ is received if every sub-function is finished within its deadline. The extended *EdgeOPT* for function chain scheduling is outlined in *Algorithm 4*. We map the sub-function to the server with the least increment of threshold value at each step. If the threshold in every step can be covered by the reward, the task is accepted and we process the task in the order of its chain dependence.

IV. Numerical Results

A. Experimental Setup

1) Network Settings: Consider a 1×1 km^2 area with randomly deployed edge clouds. All users are uniformly distributed, whose channel gains $|h_{mk}|^2$ (in dB) are modeled by $-140.7 - 36.7 \log_{10}(d_{mk}) + \psi$ [50], where d_{mk} (in km) is the distance between the user k and the edge cloud m, ψ is a Gauss-distributed random variable with zero mean and 10 dB standard deviation to capture log-normal shadowing. The noise power spectral density N_0 is -174 dBm/Hz and the SNR gap Γ is set to 7.6288 [51]. $\hat{I}_k$ is chosen such that $\underline{\mathfrak{r}}_{mk} = 0.6 \times \tau_k$. Task workload is measured by its required CPU cycles and the computation resource is measured by CPU cycle frequency. Unless otherwise specified, the bandwidth for each sub-channel is set to 2 MHz, the channel number at each base station is 10, the total computation resource of the servers is 15 G cycles/s, the maximum storage is 64 GB, and the time horizon is set to 10 seconds.

2) Task Settings: Task generation is modeled by a Poisson process, with a default arrival rate of 50. Mobile device transmission power $p_k \in [50, 150]$ mW, task packet size $s_k \in [10, 600]$ Kbit, task workload demand $w_k \in [0.01, 1]$ G cycles, memory requirement for each container type $\mathcal{D}_l \in [6, 10]$ GB [42], the start-up time $z_l \in [0.15, 0.85]$ [52], and the tolerable deadline $\tau_k \in [0.1, 1.7]$ s are all generated uniformly at random [53]. Task reward $\mathcal{R}_k \in [1, 50]$ follows a uniform distribution. The parameters γ_j and η are set to $\frac{0.01}{C_j}\mathbb{E}(\mathcal{R})$ and 20, respectively. All results are averaged over 500 Monte-Carlo simulations to reveal their average performance.

3) Compared Baselines: We compare our algorithm with the following heuristics and variations of our proposal:

- Parallel first-come-first-serve (P-FCFS): Tasks are assigned their required bandwidth for transmission as they arrive until the tasks in transmission occupy the entire available band. Transmitted tasks are stacked in a queue, in which the foremost tasks that finish container start-ups share the computation resources equally for processing.
- EdgeOPT+highest density most (EHDM): This baseline replaces the resource allocation subroutine of EdgeOPT by highest density most (HDM) method, which assigns the most urgent task the most computing resource, while other tasks are allocated with the minimum required computation resource to guarantee on-time completion.
- Greedy: The greedy heuristic accepts all tasks that satisfy the given constraints. Accepted tasks are transmitted in the specified channels and are processed using the minimum required computation resource.
- Greedy+resource allocation subroutine (Greedy+RA): The computation resources for processing the tasks are allocated using *Algorithm 2*. The feasibilities of the schedules are checked using *Algorithm 3* before the greedy acceptance of the tasks.
- Greedy+highest density most (GHDM): Greedily accepts all tasks that satisfy the given constraints, then allocate computation resources using HDM.

Note that P-FCFS is a best-effort scheduler that accepts tasks as much as it can, while the other algorithms are deadline-aware and contain rejection mechanisms based on the available computational resources.

B. Cumulative Reward and Delay Statistics

In Fig. 3, we investigate the cumulative reward as a function of time under different schemes, where only the rewards of tasks finished within their deadlines are counted. As we can observe, *EdgeOPT* outperforms all the other baselines in terms of cumulative reward, indicating that the proposed threshold structure is advantageous over the myopic greedy ones when tasks arrive online. Besides, the gain does not solely come from the threshold function, as evidenced by the fact that EHDM has approximately the same performance as Greedy + RA. Algorithms that allocate computation resource

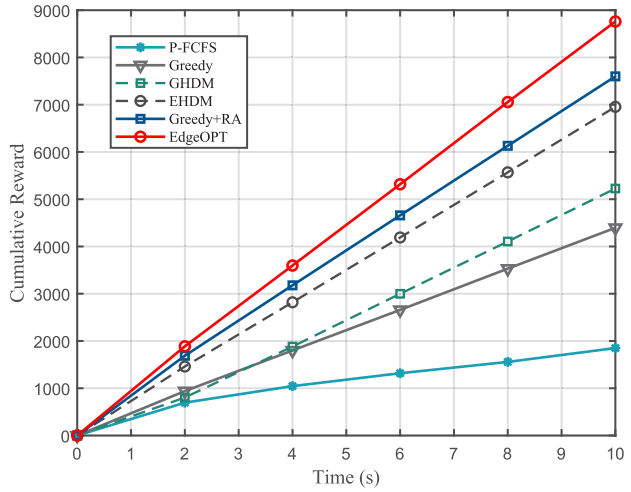

Fig. 3. Cumulative reward achieved by a single edge cloud as a function of time.

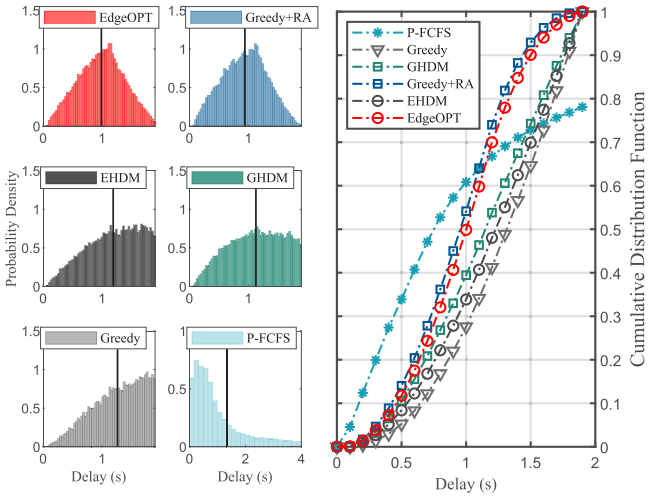

Fig. 4. Delay statistics of different methods. The black vertical lines on the left part denote the average values.

using *Algorithm 2* always outperform the ones using HDM, implying that *Algorithm 2* produces promising allocations by considering the sum remaining completion times of all tasks. Thus, we conclude that the proposed adaptive threshold collaborates well with the dynamic resource allocation subroutine, and both helps improve the achievable reward.

In Fig. 4, we display the probability density functions and the cumulative distribution functions of the delay resulting from the tasks accepted by different methods. As we can see, P-FCFS has the worst average delay of 1.331s as it does not guarantee the on-time completions. Overtime tasks lead to computing resource contention, thus resulting in a heavy-tailed delay distribution. *EdgeOPT* and Greedy + RA achieve mean delays of 0.989s and 0.942s, while EHDM and GHDM exhibit average delays of 1.189s and 1.129s, respectively. This fact implies that methods using *Algorithm 2* for resource allocation outperform those using HDM in terms of average delay. Moreover, *Algorithm 2* achieves delays below 1.5s for 92% of tasks, while HDM achieves this for only 72%. Therefore, it can be concluded that the proposed resource allocation subroutine

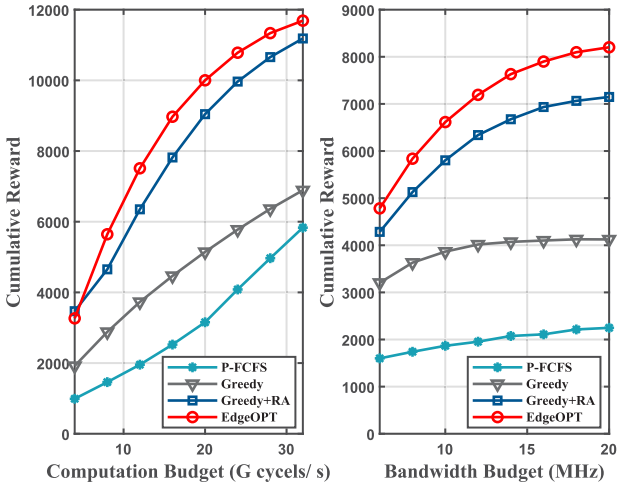

Fig. 5. Cumulative reward of a single edge cloud as a function of resource capacity.

not only improves the cumulative reward, but also reduces the average delay and the proportion of long executing tasks.

C. Impact of Resource Capacity

In Fig. 5, we study the impact of resource capacity on the cumulative reward. It can be observed that increasing either the computation resource or the available bandwidth affects positively the cumulative reward. Diminishing returns are observed when only one of the resource budget is increased, which is due to the fact that the other resource continues to act as a fundamental bottleneck for the edge cloud in completing the accepted tasks within the hard deadlines. Moreover, P-FCFS performs the worst in all tested settings, indicating that a proactive rejection mechanism can help improve the efficiency of a congested system when tasks are latency-sensitive. In general, our proposed algorithm achieves the highest cumulative reward and demonstrates superiority over all the baselines in resource-abundant situations.

D. Impact of Task Pattern

In Fig. 6, the cumulative reward under different task patterns is investigated. We respectively vary the average value of packet size, workload demand, task release rate and average deadline to obtain a spectrum of results. In Fig. 6(a), an inverse relation between cumulative reward and average packet size can be observed in all algorithms. This is due to the increment of packet size will increase the transmission time, leaving shorter time for task execution, which implies the tasks will require more computation resources in order to have a faster processing speed to be finished on time. Therefore, the server can only schedule a smaller set of tasks due to the limited computation resource. Our proposed algorithm achieves the highest cumulative reward in all tested average packet sizes, which suggests that our proposal can successfully adapt to various scenarios with different average packet sizes.

In Fig. 6(b), we study the cumulative reward as a function of the average task demand. As can be observed, the increment of average task demand decreases the cumulative reward for all algorithms, since more task with unsatisfiable demand will be rejected when the overall computation resource is limited.

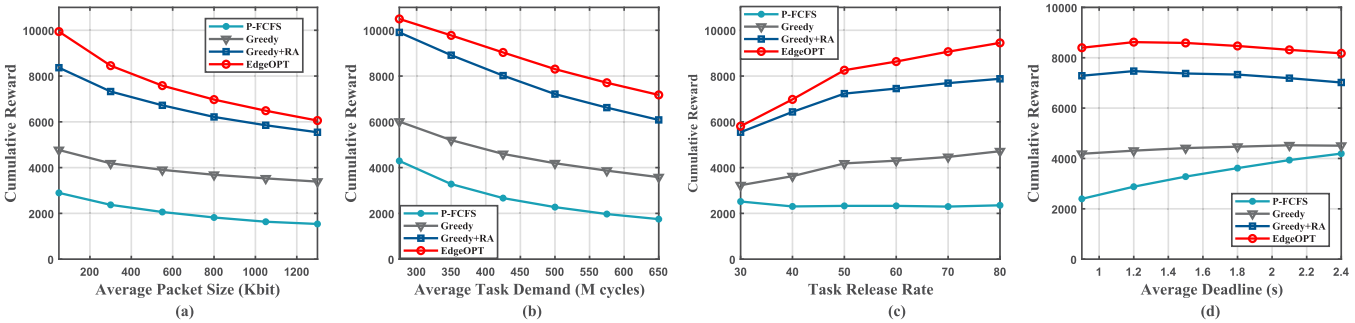

Fig. 6. Cumulative reward of a single edge cloud under different task patterns.

P-FCFS exhibits the worst performance in all tested task workload demands due to the resource contention resulting from the unsatisfiable tasks. Overall, *EdgeOPT* achieves the highest reward no matter how the average task demand changes.

Fig. 6(c) shows the dependence of cumulative reward on average task release rate, which is defined as the average number of task arrivals per second. The proposed algorithm exhibits superiority in all tested task release rates, with an increasing advantage in task-intensive situation. This attributes to the threshold structure of our proposal that reserves resources for more computation-demanding tasks, which caters to task-intensive situations where it is more likely to generate tasks with higher rewards, thereby providing a greater advantage.

In Fig. 6(d), we show the cumulative reward as a function of the average deadline. A slight increment of cumulative reward is observed when the average deadline is below 1.2s. Further extending the deadlines results in accepting more satisfiable tasks once the system has vacant resource. However, this causes resource contention, resulting in a higher number of tasks being completed just before their deadlines. Consequently, longer deadlines translate to extended server occupancy periods, thereby slightly reducing the cumulative reward. Reducing the deadlines has the worst impact on the P-FCFS algorithm, where more overtime tasks negatively affect the system when the deadlines are more urgent. This result highlights the importance of the deadline-aware resource allocation, which allows dynamic resource redistributions in every task states to improve the system throughput. We can conclude from Figs. 6(a)-(d) that our proposal provides the highest cumulative reward across all settings, which demonstrates its robustness and adaptivity to various task patterns.

E. Impact of Transmission Rate Misestimation

In Fig. 7, we investigate the robustness of *EdgeOPT* in face of inaccurate estimation of transmission rate, which may result from the unpredictable interference or the time varying channel gain during the offloading process. Transmission rate reduction due to this effect is modeled by the misestimation factor ε_r^{max}. Define ε_k as the percentage of transmission rate loss of user k, which is chosen uniformly at random within the range $[0, \varepsilon_r^{max}]$ to obtain the real transmission rate $\bar{r}_{mkt} = (1-\varepsilon_k) r_{mkt}$. We vary ε_r^{max} from 0 to 0.35 to obtain results representing different degrees of misestimation. We observe

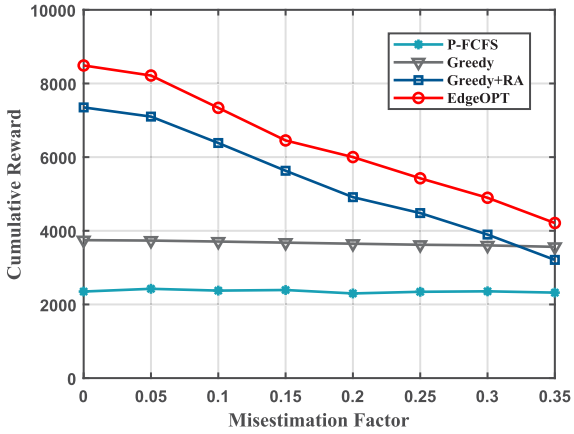

Fig. 7. Cumulative reward as a function of rate misestimation factor.

that Greedy + RA maintains its advantage over Greedy up to a misestimation factor of 0.32. This can be attributed to the proposed resource allocation subroutine that dynamically adjusts the computing resources according to the remaining workload for on-time task completion. Overall, our proposed method consistently achieves the highest cumulative reward in all tested misestimation ratios, demonstrating that *EdgeOPT* is robust against inaccurate transmission rate estimation.

F. Impact of Edge Cloud Number

We investigate the impact of edge cloud number in Fig. 8, where we show the dependence of cumulative reward on server number on the left, and the differences between *EdgeOPT* and the baselines are displayed on the right. The time horizon is set to 5s. It can be observed that deploying more edge clouds increases the cumulative reward due to the increment in more computation and communication resources. The reward achieved by Greedy grows at the lowest rate, and is outperformed by P-FCFS when the server number exceeds 4. Sub-linear growths of cumulative reward are also observed, as the inter-cell interference counteracts the return brought by the growing server numbers. The proposed *EdgeOPT* algorithm mitigates this effect by proactively scheduling the transmission channels and execution servers, thus outperforming all the baselines across the tested server numbers.

When the number of servers is sufficiently high, P-FCFS, Greedy, and Greedy + RA may achieve performances close to *EdgeOPT*, since further deploying servers will relieve the

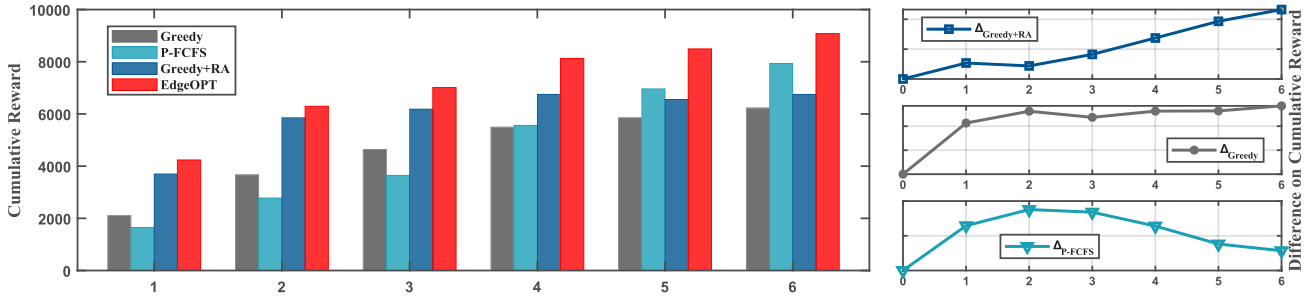

Fig. 8. Cumulative reward of multiple edge clouds as a function of server number.

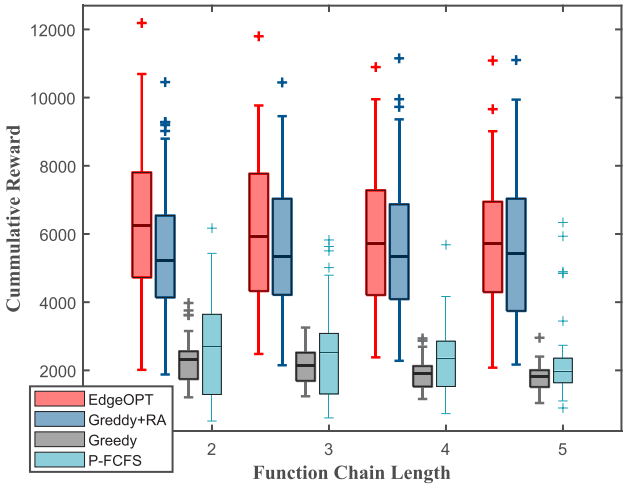

Fig. 9. Cumulative reward as a function of function chain length.

task intensity on each edge cloud. The time-complexities for scheduling each task using these methods are $\mathcal{O}(1)$, $\mathcal{O}(|\mathcal{M}||\mathcal{C}|)$, and $\mathcal{O}\left(|\mathcal{M}|(|\mathcal{S}_t^m|^2+|\mathcal{C}|\right)$, respectively. Here, $|\mathcal{S}_t^m|$ is the number of tasks the server is currently handling. Thus, low complexity alternatives might be more preferable when the network has abundant processing capacity. Overall, the proposed *EdgeOPT* algorithm consistently achieves the highest cumulative reward in all numbers of edge clouds, therefore we conclude that the proposed method improves the cumulative reward for edge computing system with multiple edge clouds under heavy load scenarios.

G. Function Chain Scheduling

In Fig. 9, we investigate the cumulative reward under different service function lengths. The time horizon is set to 5s, and the server number is set to 3. As we can observe from the boxplot, Greedy achieves the lowest cumulative reward, while P-FCFS exhibits better but more scattered reward. This is due to Greedy schedules tasks more conservatively by taking into account all the deadline-related constraints, whereas P-FCFS is a best-effort scheduler that depends more on the specific task instance. Across all tested function chain lengths, the proposed *EdgeOPT* algorithm achieves the highest cumulative reward in terms of both the median value and the 25-percentile, implying that *EdgeOPT* is both efficient and steady. Such superiority is attributed to the dedicated threshold structure, which enables mapping each sub-function to a suitable server to obtain the lowest scheduling cost for each online task.

V. Conclusion

In this paper, we proposed a competitive algorithm, namely *EdgeOPT*, for online parallel task scheduling in edge computing, aiming to maximize the cumulative reward of tasks subject to their hard deadlines. *EdgeOPT* obtains scheduling decisions leveraging an adaptive threshold structure at each edge cloud, and allows efficient resource allocation by exploiting the inherent water-filling structure. We proved a bounded competitive ratio for *EdgeOPT* in monolithic task scheduling case, and proposed an efficient algorithm for function chain scheduling. Extensive experiments showed that our proposal outperforms all the baselines, indicating that it is promising for practical systems.

Appendix A
Proof of Lemma 2

By the partition rule, the duration for consecutive non-decreasing *receiving slots* is no more than $\tau_{\max}$. On the other hand, at any time t, all accepted tasks will be completed before $t+\tau_{\max}$, after which $\rho_j^m(S_t^m)$ will return to 0. Since all the tasks arrive when $\rho_j^m(S_t^m)=0$ will be accepted, the segment of time between $t+\tau_{\max}$ and the start of the next *receiving slot* can be discarded without affecting the analysis. Therefore, the duration of any interval can be upper-bounded by the sum of the two parts, which is $2\tau_{\max}$. ■

Appendix B
Proof of Lemma 3

Observe that under any resource allocation policy, there are at most $\lfloor \frac{1}{\epsilon'} \rfloor \le \frac{1}{\epsilon'}$ tasks existing on a server simultaneously. Given full computation resource, the completion time $t_{k'}$ of any task $k' \in \mathcal{K}$ is $\left(v_{mk}+\frac{w_{k'}}{\mathcal{F}_{max}}\right)$. Based on this, we construct an impossible instance where tasks arrive in $|\mathcal{G}|$ consecutive groups. Each group $g_l \in \mathcal{G}$ contains $\frac{1}{\epsilon'}$ replicas of task k' that arrive simultaneously and are permitted with full computation resource. Once a group is completed, another group arrives instantly. Denote t_g the completion time of a group, it follows that:

$$
\begin{aligned}
|\mathcal{G}| &= \left\lceil \frac{2\tau_{\max}}{t_g} \right\rceil < 1+\frac{2\tau_{\max}}{t_g} = 1+\frac{2\tau_{\max}}{v_{mk}+\frac{w_{k'}}{\mathcal{F}^m}} \\
&= 1+2\tau_{\max}\left[v_{mk}+\left(\frac{w_{k'}}{\tau_{k'}-v_{mk}}\right)\left(\frac{\tau_{k'}-v_{mk}}{\mathcal{F}^m}\right)\right]^{-1} \\
&= 1+2\tau_{\max}\left[v_{mk}+q_{k'F}^m\left(\tau_{k'}-v_{mk}\right)\right]^{-1}
\end{aligned}
$$

$$\leq 1+2\tau_{\max}\left[v_{mk}+\epsilon'\left(\tau_{k'}-v_{mk}\right)\right]^{-1}$$
$$=1+\frac{2\tau_{\max}}{\left(1-\epsilon'\right)v_{mk}+\epsilon'\tau_{k'}}\leq 1+\frac{2\tau_{\max}}{\epsilon'\tau_{k'}}$$
$$\leq 1+\frac{2\lambda}{\epsilon'}, \tag{39}$$

where the first equality is due to *Lemma 1*, the second inequality is because $\epsilon'\leq\frac{q^m_{k'F}}{\mathcal{F}_{max}}=\frac{w_{k'}}{(\tau_{k'}-v_{mk})\mathcal{F}_{max}}$, the third inequality is from $v_{mk}>0,\epsilon'<1$, and the last inequality is due to $\tau_{min}\leq\tau_{k'}$.

Decompose the tasks accepted during interval T into different groups $g_l\in\mathcal{G}$, we have:

$$\sum_{i\in\hat{O}_T}q^m_{ij}\leq\sum_{l=1}^{|\mathcal{G}|}\sum_{i\in g_l}q^m_{ij}$$
$$\leq\left(1+\frac{2\lambda}{\epsilon'}\right)\sum_{i\in g_l}C^m_j$$
$$\leq\left(1+\frac{2\lambda}{\epsilon'}\right)\frac{1}{\epsilon'}C^m_j=\left(\frac{1}{\epsilon'}+\frac{2\lambda}{\epsilon'^2}\right)C^m_j, \tag{40}$$

where the first inequality is because not all tasks in the impossible instance can be accepted, the second inequality is due to the upper bound on $|\mathcal{G}|$, and the last inequality is because there are at most $\frac{1}{\epsilon'}$ tasks can be served simultaneously. ■

Appendix C
Proof of Lemma 4

In *Case 1*, there is still enough capacity for every task at any time. Thus these tasks were rejected only because their reward did not exceed the acceptance threshold. It holds that:

$$f(\hat{O}^m_T\setminus\tilde{S}^m_T)=\sum_{i\in\hat{O}^m_T\setminus\tilde{S}^m_T}\mathcal{R}_i$$
$$\leq\sum_{i\in\hat{O}^m_T\setminus\tilde{S}^m_T}\sum_{j\in\mathcal{J}}\gamma^m_j\left(e^{\rho^m_j(S^*_{m,T})}-1\right)q^m_{ij}$$
$$=\sum_{j\in\mathcal{J}}\left[\gamma^m_j\left(e^{\rho^m_j(S^*_{m,T})}-1\right)\sum_{i\in\hat{O}^m_T\setminus\tilde{S}^m_T}q^m_{ij}\right]$$
$$\leq\left(\frac{1}{\epsilon'}+\frac{2\lambda}{\epsilon'^2}\right)\sum_{j\in\mathcal{J}}\gamma^m_jC^m_j\left(e^{\rho^m_j(S^*_{m,T})}-1\right), \tag{41}$$

where the last inequality is due to *Lemma 3* and the fact that $\hat{O}^m_T\setminus\tilde{S}^m_T\subseteq\hat{O}^m_T$. ■

Appendix D
Proof of Lemma 6

In *Case 2*, we upper bound the offline optimum by focusing on the contributions from dimensions in $\mathcal{J}^T_1\cup\mathcal{J}'_T$.

$$f(\hat{O}^m_T)=\sum_{i\in\hat{O}^m_T}\mathcal{R}_i$$
$$=\sum_{i\in\hat{O}^m_T}\frac{\mathcal{R}_i}{\sum_{j\in\mathcal{J}^T_1\cup\mathcal{J}'_T}\gamma_jq_j(\boldsymbol{\varphi}_i,a_i)}$$
$$\times\sum_{j\in\mathcal{J}^T_1\cup\mathcal{J}'_T}\gamma_jq_j(\boldsymbol{\varphi}_i,a_i)$$
$$\leq\sum_{i\in\hat{O}^m_T}\frac{\mathcal{R}_i}{\sum_{j\in\mathcal{J}^T_1\cup\mathcal{J}'_T}\gamma_jq_j(\boldsymbol{\varphi}_i,a_i)}\sum_{j\in\mathcal{J}^T_1\cup\mathcal{J}'_T}\gamma_jq^m_{ij}$$
$$\leq\sigma'\sum_{j\in\mathcal{J}^T_1\cup\mathcal{J}'_T}\gamma_j\sum_{i\in\hat{O}^m_T}q^m_{ij}$$
$$\leq\sigma'\left(\frac{1}{\epsilon'}+\frac{2\lambda}{\epsilon'^2}\right)\sum_{j\in\mathcal{J}^T_1\cup\mathcal{J}'_T}\gamma_jC^m_j, \tag{42}$$

where the first inequality is due to the definition of σ' and the third inequality is due to *Lemma 3*. ■

Appendix E
Proof of Proposition 1 under Case 2

We partition the constraint set $\mathcal{J}^T_1\cup\mathcal{J}'_T$ into $\mathcal{J}^T_1\setminus\left(\mathcal{J}^T_2\cup\mathcal{J}'_T\right)$, $\mathcal{J}^T_2\setminus\mathcal{J}'_T$, $\mathcal{J}'_T\cap\{j:H^{m,j}_T=1\}$ and $\mathcal{J}'_T\setminus\{j:H^{m,j}_T=1\}$, whose contributions to the objective are lower bounded by $\mathcal{L}_{T,3}$, $\mathcal{L}_{T,4}$, $\mathcal{L}_{T,5}$, and $\mathcal{L}_{T,6}$, respectively. The same idea as *Case 1* gives:

$$f(\tilde{S}^m_T)\geq\sum_{i\in\tilde{S}^m_T}\sum_{j\in\mathcal{J}^T_1\cup\mathcal{J}'_T}\gamma_jq^m_{ij}\left(e^{\left\lfloor\frac{u_j(S^m_{T,i-1})}{C^m_j}\ln(\theta^m_i\eta)\right\rfloor}-1\right)$$
$$=\sum_{j\in\mathcal{J}^T_1\cup\mathcal{J}'_T}\sum_{n=\rho_j(S'_{m,T-1})}^{\rho_j(S^*_{m,T})}(e^n-1)$$
$$\times\left[\sum_{i\in S^{m,j}_{T,n}\setminus S^{m,j}_{T,n-1}}\gamma_jq^m_{ij}\right]$$
$$\geq\mathcal{L}'_{T,6}+\mathcal{L}(\mathcal{J}^T_1\setminus\left(\mathcal{J}^T_2\cup\mathcal{J}'_T\right))+\mathcal{L}\left(\mathcal{J}^T_2\setminus\mathcal{J}'_T\right)$$
$$+\mathcal{L}\left(\mathcal{J}'_T\setminus\{j:H^{m,j}_T=1\}\right), \tag{43}$$

where $\mathcal{L}'_{T,6}$ is the sum term for $j\in\mathcal{J}'_T\cap\{j:H^{m,j}_T=1\}$ in the second line of (43). A key observation is that $j\in\mathcal{J}'_T\cap\{j:H^{m,j}_T\}$ implies $\rho_j(S'_{m,T-1})=\rho_j(S^*_{m,T})$, and such case is equivalent to $j\in\mathcal{J}\setminus\mathcal{J}^T_1$ under *Case 1*, therefore yielding a similar form of the lower bound:

$$\mathcal{L}'_{T,6}=\sum_{j\in\mathcal{J}^T_1\cup\mathcal{J}'_T}\left(e^{\rho_j(S^*_{m,T})}-1\right)\sum_{i\in S^{m,j}_{T,0}}\gamma_jq^m_{ij}$$
$$\geq\sum_{j\in\mathcal{J}'_T\cap\{j:H^{m,j}_T=1\}}(e^{\rho_j(S^*_{m,T})}-1)\epsilon'\gamma_jC^m_j$$
$$\geq\sum_{j\in\mathcal{J}'_T\cap\{j:H^{m,j}_T=1\}}(\frac{\theta^{[n]}\eta}{e^2}-1)\epsilon'\gamma_jC^m_j\overset{\text{def}}{=}\mathcal{L}_{T,6}, \tag{44}$$

where the second inequality is because $\rho_j(S^*_{m,T})\geq\left\lfloor\ln(\theta^{[n]}\eta)\right\rfloor-1>\ln(\theta^{[n]}\eta)-2$.

For $j\in\mathcal{J}_1\setminus\mathcal{J}'_T$, we have $u_j(S^*_{m,T})<(1-\epsilon)C^m_j$, which is similar to as *Case 1*. Thus we have similar lower bounds:

$$\mathcal{L}_{T,3}=[e-1]\sum_{j\in\mathcal{J}^T_1\setminus(\mathcal{J}^T_2\cup\mathcal{J}'_T)}\epsilon'\gamma_jC^m_j,$$
$$\mathcal{L}_{T,4}=\frac{e^2-2(e-1)-1}{e-1}\sum_{j\in\mathcal{J}^T_2\setminus\mathcal{J}'_T}\gamma_jC^m_j\left[\frac{1}{\ln(\theta^{[n]}\eta)}-\epsilon\right].$$

For $j \in \mathcal{J}'_T \setminus \{j : H_T^{m,j} = 1\}$, we have $H_T^{m,j} \geq 2$. Since $C_j^m - u_j(S^*_{m,T}) \leq \epsilon C_j^m$, we have: $u_j(S^*_{m,T}) \geq (1-\epsilon)C_j^m$. Taking this into $\rho_j(S^*_{m,T}) = \left\lfloor \frac{u_j(S^*_{m,T})}{C_j^m} \ln(\theta^{[n]}\eta) \right\rfloor$ gives: $\rho_j^m(S^*_{m,T}) \geq \lfloor (1-\epsilon)\ln(\theta^{[n]}\eta) \rfloor \geq \lfloor \ln(\theta^{[n]}\eta) - \delta \rfloor \geq \lfloor \ln(\theta^{[n]}\eta) - 1 \rfloor \geq \lfloor \ln(\theta^{[n]}\eta) \rfloor - 1$. Combining these gives:

$$\begin{aligned}
&\mathcal{L}\left(\mathcal{J}'_T \setminus \{j : H_T^j = 1\}\right) \\
&= \sum_{j \in \mathcal{J}'_T \setminus \{j: H_T^{m,j}=1\}} \gamma_j \sum_{n=0}^{\Delta_{t,j}^m(S^*_{m,T})} (e^n - 1) \sum_{i \in S_{T,n}^{m,j} \setminus S_{T,n-1}^{m,j}} q_{ij}^m \\
&\geq \sum_{j \in \mathcal{J}'_T \setminus \{j: H_T^{m,j}=1\}} \gamma_j \sum_{h=0}^{H_T^{m,j}-1} \left(e^h - 1\right) \left[\sum_{i \in G_{T,h}^{m,j} \setminus G_{T,h-1}^{m,j}} q_{ij}^m \right] \\
&\geq \sum_{j \in \mathcal{J}'_T \setminus \{j: H_T^{m,j}=1\}} \gamma_j \left[\sum_{i \in G^{m,j}_{T,H_T^j-1} \setminus G_{T,0}^{m,j}} q_{ij}^m \right],
\end{aligned} \tag{45}$$

where the last inequality is due to $H_T^{m,j} \geq 2$ and the fact that $e^h - 1 > 1$ when $h \geq 1$. For all $j \in \mathcal{J}'_T$, we have $\sum_{i \in S^*_{m,T}} q_j(\boldsymbol{\varphi}_i, a_i) = \sum_{i \in G^{m,j}_{T,H_T^j-1}} q_j(\boldsymbol{\varphi}_i, a_i) > (1-\epsilon)C_j^m$. Moreover, the piece-wise constant property of $\rho_j^m(S)$ gives $\sum_{i \in G_{T,0}^{m,j}} q_j(\boldsymbol{\varphi}_i, a_i) \leq \epsilon C_j^m + \frac{C_j^m}{\ln(\eta)}$, where the inequality is due to the fact that $\theta^{[n]}$ is no less than 1. Combining these yields:

$$\begin{aligned}
\sum_{i \in G^{m,j}_{T,H_T^j-1} \setminus G_{T,0}^{m,j}} q_{ij}^m &= \sum_{i \in G^{m,j}_{T,H_T^j-1} \setminus G_{T,0}^{m,j}} q_j(\boldsymbol{\varphi}_i, a_i) \\
&\quad + \sum_{i \in G^{m,j}_{T,H_T^j-1} \setminus G_{T,0}^{m,j}} \mathbb{1}_{q_j(\boldsymbol{\varphi}_i, a_i)=0} \\
&\geq \sum_{i \in G^{m,j}_{T,H_T^j-1}} q_j(\boldsymbol{\varphi}_i, a_i) \\
&\quad - \sum_{i \in G_{T,0}^{m,j}} q_j(\boldsymbol{\varphi}_i, a_i) \\
&\geq (1-\epsilon)C_j^m - \left(\epsilon C_j^m + \frac{C_j^m}{\ln(\eta)}\right) \\
&= \left(1 - 2\epsilon - \frac{1}{\ln(\eta)}\right) C_j^m.
\end{aligned} \tag{46}$$

Taking (46) into (45) gives the lower bound $\mathcal{L}_{T,5}$:

$$\begin{aligned}
&\mathcal{L}\left(\mathcal{J}'_T \setminus \{j : H_T^{m,j} = 1\}\right) \\
&> \sum_{j \in \mathcal{J}'_T \setminus \{j: H_T^{m,j}=1\}} \gamma_j^m \left(1 - 2\epsilon - \frac{1}{\ln(\eta)}\right) C_j^m \stackrel{\text{def}}{=} \mathcal{L}_{T,5}.
\end{aligned} \tag{47}$$

Since we always choose η that satisfies $\epsilon < \frac{1}{2} - \frac{1}{2\ln(\eta)}$, every term in (47) is positive. Therefore, we can apply *Fact 1*:

$$\frac{f(\hat{O}_T^m)}{f(\tilde{S}_T^m)} \leq \frac{\sigma'\left(1 + \frac{\lambda}{\epsilon'}\right) \sum_{j \in \mathcal{J}_1^T \cup \mathcal{J}'_T} \gamma_j C_j^m}{\mathcal{L}_{T,6} + \mathcal{L}_{T,3} + \mathcal{L}_{T,4} + \mathcal{L}_{T,5}}. \tag{48}$$

Finally, since (i) for $\forall j \in \mathcal{J}$, $\epsilon \ln(\theta^{[m]}\eta) < 1$, we have $1 - \epsilon\ln(\theta^{[m]}\eta) = 1 - \delta > 0$ and (ii) by $1 - 2\epsilon - \frac{1}{\ln(\eta)} > 0$, we let $c' \stackrel{\text{def}}{=} (1-2\epsilon)\ln(\eta) - 1 > 0$, the result can be concluded as:

$$\begin{aligned}
\frac{f(\hat{O}_T^m)}{f(\tilde{S}_T^m)} &\leq \sigma'\left(\frac{1}{\epsilon'} + \frac{2\lambda}{\epsilon'^2}\right) \max\left\{ \frac{e^2}{(\theta^{[m]}\eta - e^2)\epsilon'}, \right. \\
&\qquad \left. \frac{1}{(e-1)\epsilon'}, \frac{(e-1)\ln(\theta^{[m]}\eta)}{(e^2 - 2(e-1) - 1)(1-\delta)}, \frac{\ln(\theta^{[m]}\eta)}{c'} \right\} \\
&= \mathcal{O}(\lambda \ln(\theta^{[m]})) \leq \mathcal{O}(\lambda \theta^{[m]} \ln(\theta^{[m]})),
\end{aligned} \tag{49}$$

which completes the proof. ■

ACKNOWLEDGMENT

The authors would like to thank the editors and the anonymous reviewers, whose invaluable comments helped improve the presentation of this article substantially.

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, 4th Quart., 2017.

[2] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[3] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2015.

[4] Z. Kuai, T. Wang, and S. Wang, "Fair virtual network function mapping and scheduling using proximal policy optimization," *IEEE Trans. Commun.*, vol. 70, no. 11, pp. 7434–7445, Nov. 2022.

[5] Z. Kuai and S. Wang, "Online virtual network function scheduling towards deterministic latency," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Kuala Lumpur, Malaysia, Dec. 2023, pp. 249–254.

[6] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

[7] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[8] A. Ben-Ameur, A. Araldo, and T. Chahed, "Multiple resource allocation in multi-tenant edge computing via sub-modular optimization," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Rome, Italy, Jun. 2023, pp. 3738–3743.

[9] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 668–682, Mar. 2019.

[10] Z. Wang, H. Du, and Q. Ye, "HTR: A joint approach for task offloading and resource allocation in mobile edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Montreal, QC, Canada, Jun. 2021, pp. 1–6.

[11] M.-H. Chen, B. Liang, and M. Dong, "Multi-user multi-task offloading and resource allocation in mobile cloud systems," *IEEE Wireless Commun.*, vol. 17, no. 10, pp. 6790–6805, Oct. 2018.

[12] S. Misra and N. Saha, "Detour: Dynamic task offloading in software-defined fog for IoT applications," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1159–1166, May 2019.

[13] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE Trans. Netw.*, vol. 26, no. 6, pp. 2651–2664, Dec. 2018.

[14] T. Ji et al., "Energy-efficient computation offloading in mobile edge computing systems with uncertainties," *IEEE Trans. Wireless Commun.*, vol. 21, no. 8, pp. 5717–5729, Aug. 2022.

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

[16] S. Jošilo and G. Dán, "Joint wireless and edge computing resource management with dynamic network slice selection," *IEEE/ACM Trans. Netw.*, vol. 30, no. 4, pp. 1865–1878, Aug. 2022.

[17] X. Wang, J. Ye, and J. C. S. Lui, "Decentralized task offloading in edge computing: A multi-user multi-armed bandit approach," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, London, U.K., May 2022, pp. 1199–1208.

[18] R. Zhou et al., "Online task offloading for 5G small cell networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 2103–2115, Jun. 2022.

[19] Z. Sun and M. R. Nakhai, "An online learning algorithm for distributed task offloading in multi-access edge computing," *IEEE Trans. Signal Process.*, vol. 68, pp. 3090–3102, 2020.

[20] X. Pang, Z. Wang, J. Li, R. Zhou, J. Ren, and Z. Li, "Towards online privacy-preserving computation offloading in mobile edge computing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, London, U.K., May 2022, pp. 1179–1188.

[21] Q. Fan, J. Bai, H. Zhang, Y. Yi, and L. Liu, "Delay-aware resource allocation in fog-assisted IoT networks through reinforcement learning," *IEEE Internet Things J.*, vol. 9, no. 7, pp. 5189–5199, Apr. 2022.

[22] T. Liu, Y. Zhang, Y. Zhu, W. Tong, and Y. Yang, "Online computation offloading and resource scheduling in mobile-edge computing," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6649–6664, Apr. 2021.

[23] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.

[24] B. Lucier et al., "Efficient online scheduling for deadline-sensitive jobs," in *Proc. ACM SPAA*, Montreal, QC, Canada, Jul. 2013, pp. 305–314.

[25] Z. Han, H. Tan, X.-Y. Li, S. H.-C. Jiang, Y. Li, and F. C. M. Lau, "OnDisc: Online latency-sensitive job dispatching and scheduling in heterogeneous edge-clouds," *IEEE/ACM Trans. Netw.*, vol. 27, no. 6, pp. 2472–2485, Dec. 2019.

[26] J. Meng, H. Tan, X.-Y. Li, Z. Han, and B. Li, "Online deadline-aware task dispatching and scheduling in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1270–1286, Jun. 2020.

[27] C. Zhang et al., "Online approximation scheme for scheduling heterogeneous utility jobs in edge computing," *IEEE/ACM Trans. Netw.*, vol. 31, no. 1, pp. 352–365, Feb. 2023.

[28] J. S. Chadha, N. Garg, A. Kumar, and V. N. Muralidhara, "A competitive algorithm for minimizing weighted flow time on unrelatedmachines with speed augmentation," in *Proc. ACM STOC*, Bethesda, MD, USA, May/Jun. 2009, pp. 679–684.

[29] Z. Zheng and N. B. Shroff, "Online multi-resource allocation for deadline sensitive jobs with partial values in the cloud," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.

[30] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 945–953.

[31] J. Zhu and S. Wang, "Delay-guaranteed resource allocation for deterministic communications: An efficient stochastic network calculus method," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Kuala Lumpur, Malaysia, Dec. 2023, pp. 7061–7066.

[32] I.-H. Hou et al., "Asymptotically optimal algorithm for online reconfiguration of edge-clouds," in *Proc. ACM MobiHoc*, Paderborn, Germany, 2016, pp. 291–300.

[33] A. U. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 393–413, 1st Quart., 2013.

[34] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Trans. Veh. Technol.*, vol. 66, no. 4, pp. 3435–3447, Apr. 2017.

[35] A. J. Goldsmith and S.-G. Chua, "Variable-rate variable-power MQAM for fading channels," *IEEE Trans. Commun.*, vol. 45, no. 10, pp. 1218–1230, Oct. 1997.

[36] Q. Ye, B. Rong, Y. Chen, M. Al-Shalash, C. Caramanis, and J. G. Andrews, "User association for load balancing in heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 6, pp. 2706–2716, Jun. 2013.

[37] Y. Qu et al., "Robust offloading scheduling for mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 7, pp. 2581–2595, Jul. 2022.

[38] Z. Nan, S. Zhou, Y. Jia, and Z. Niu, "Joint task offloading and resource allocation for vehicular edge computing with result feedback delay," *IEEE Trans. Wireless Commun.*, vol. 22, no. 10, pp. 6547–6561, Oct. 2023.

[39] P. Barham et al., "Xen and the art of virtualization," in *Proc. ACM SOSP*, Boulton Landing, NY, USA, Oct. 2003, pp. 1–14.

[40] S. Chen, L. Wang, and F. Liu, "Optimal admission control mechanism design for time-sensitive services in edge computing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, London, U.K., May 2022, pp. 1169–1178.

[41] T. He, H. Khamfroush, S. Wang, T. La Porta, and S. Stein, "It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Vienna, Austria, Jul. 2018, pp. 365–375.

[42] W. Chu, X. Jia, Z. Yu, J. C. S. Lui, and Y. Lin, "Joint service caching, resource allocation and task offloading for MEC-based networks: A multi-layer optimization approach," *IEEE Trans. Mobile Comput.*, vol. 23, no. 4, pp. 2958–2975, Apr. 2024.

[43] S. Li et al., "Golgi: Performance-aware, resource-efficient function scheduling for serverless computing," in *Proc. ACM SoCC*, Santa Cruz, CA, USA, Oct./Nov. 2023, pp. 32–47.

[44] R. Iyer and J. Bilmes, "Submodular optimization with submodular cover and submodular knapsack constraints," in *Proc. NeurIPS*, Lake Tahoe, NV, USA, Dec. 2013, pp. 1–9.

[45] L. Yang et al., "Competitive algorithms for online multidimensional knapsack problems," in *Proc. ACM SIGMETRICS*, Mumbai, India, Jun. 2022, pp. 87–88.

[46] B. Sun et al., "The online knapsack problem with departures," in *Proc. ACM SIGMETRICS'23*, Orlando, FL, USA, Jun. 2023, pp. 59–60.

[47] A. Zeynali et al., "Data-driven competitive algorithms for online knapsack and set cover," in *Proc. AAAI*, Vancouver, BC, Canada, Feb. 2021, pp. 1–9.

[48] P. He, L. Zhao, S. Zhou, and Z. Niu, "Water-filling: A geometric approach and its application to solve generalized radio resource allocation problems," *IEEE Trans. Wireless Commun.*, vol. 12, no. 7, pp. 3637–3647, Jul. 2013.

[49] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge Univ. Press, 2004.

[50] S. Wang, W. Zhao, and C. Wang, "Budgeted cell planning for cellular networks with small cells," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4797–4806, Oct. 2015.

[51] W. Zhao, S. Wang, C. Wang, and X. Wu, "Approximation algorithms for cell planning in heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 1561–1572, Feb. 2017.

[52] Q. Tang et al., "Distributed task scheduling in serverless edge computing networks for the Internet of Things: A learning approach," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 19634–19648, Oct. 2022.

[53] R. Grandl et al., "Multi-resource packing for cluster schedulers," in *Proc. ACM SIGCOMM*, Chicago, IL, USA, Aug. 2014, pp. 1–12.

Yuchen Yang received the B.S. degree in physics from the Huazhong University of Science and Technology, Wuhan, China, in 2022. He is currently pursuing the Ph.D. degree with the School of Electronic Science and Engineering, Nanjing University, Nanjing, China. His research interests include machine learning and online optimization.

Shaowei Wang (Senior Member, IEEE) received the Ph.D. degree from Wuhan University, Wuhan, China, in 2006. In 2006, he joined the School of Electronic Science and Engineering, Nanjing University, Nanjing, China, as a Faculty Member, where he is currently a Full Professor. From 2012 to 2013, he was a Visiting Scholar/a Professor with Stanford University, Stanford, CA, USA, and The University of British Columbia, Vancouver, BC, Canada. His research interests include communications and networking, operations research, and machine learning.