Decentralized Scheduling and Dynamic Pricing for Edge Computing: A Mean Field Game Approach

Xiong Wang, *Member, IEEE*, Jiancheng Ye, *Member, IEEE, ACM*, and John C.S. Lui, *Fellow, IEEE, ACM*

***Abstract*— Edge computing provides a platform facilitating edge servers to contribute to computation offloading while economizing their resources. Traditional offloading solutions are mostly centralized, which are unscalable for *large-scale* edge computing networks due to complex interactions among many edge servers. Meanwhile, dynamic pricing for an operator is equally, if not more, important to accommodate users' time-varying demands for computing services. In this paper, we develop a *decentralized online optimization framework* to jointly minimize the server's cost of workload scheduling while maximizing the operator's utility of service pricing. Specifically, we employ the *mean field game* to model the collective scheduling behavior of all edge servers, thereby enabling optimal decision making only based on the server's local information. Considering the service price in practice is not adjusted as frequently as the scheduling process, we establish a *two-timescale* optimization framework, where workload scheduling at a small timescale is tightly embedded into service pricing at a large timescale. Using mean field approximation, we derive the *closed-form* expression for the minimum scheduling cost, and the approximation error is $O\left(\frac{1}{\sqrt{M}}\right)$ which declines as the number of edge servers M increases. By characterizing the influence of workload scheduling on dynamic pricing, we transform the complex service utility maximization into a succinct but equivalent problem, and thus we can make use of Lyapunov optimization to determine the optimal price over time. Extensive evaluations validate the effectiveness and optimality of our scheduling and pricing schemes.**

***Index Terms*— Edge computing, decentralized scheduling, mean field game, dynamic pricing.**

Manuscript received 18 April 2021; revised 6 May 2022; accepted 11 August 2022; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Shakkottai. Date of publication 26 September 2022; date of current version 16 June 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62202185, in part by the Research Grants Council (RGC) under Grant GRF 14200321, and in part by The Chinese University of Hong Kong (CUHK) under Grant 6905407. *(Corresponding author: Jiancheng Ye.)*

Xiong Wang is with the National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Laboratory and the Cluster and Grid Computing Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: xiongwang@hust.edu.cn).

Jiancheng Ye is with the Network Technology Laboratory and the Hong Kong Research Center, Huawei, Hong Kong (e-mail: yejiancheng@huawei.com).

John C.S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: cslui@cse.cuhk.edu.hk).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2022.3204698, provided by the authors.

Digital Object Identifier 10.1109/TNET.2022.3204698

I. INTRODUCTION

IN RECENT years, people have witnessed the unprecedented growth of data generated from the network edge, and this growth will continue with the development of 5G and Internet of Things [1]. Traditional cloud computing may suffer from high latency and service failure when there is a network connectivity problem. To reduce the impact of these issues, the edge computing paradigm has been proposed to push the cloud frontier to the edge, thereby supporting a variety of compute-intensive yet latency-sensitive applications, e.g., VR in interactive games [2], automated driving in intelligent transportation [3], and pattern recognition for object detection [4].

The key idea of edge computing is to allow users to offload their computation workloads to nearby edge servers, such as small-cell base stations or Wi-Fi access points. Compared to the cloud datacenter, edge servers only have limited computing capacity [5]. Therefore, it is essential for these servers to further offload the workloads exceeding their capacity to the remote cloud, thus leading to a hierarchical offloading structure [6] among end users, edge servers and cloud datacenter, as shown in Fig. 1. In particular, each server needs to determine the amount of workloads to be processed locally, and as well as amount to be offloaded remotely so as to minimize the scheduling cost. Nevertheless, workloads in such scheduling scheme may be aggressively processed by low-cost servers under uneven workload distribution [7], which would worsen the system performance due to large latency at the overloaded servers. As a result, server-side *load balancing* is also indispensable for ensuring balanced workload scheduling and meanwhile enhancing the fairness in cloud resource sharing [8].

Along with users' computation offloading, a service price is charged by an operator (like a content delivery network provider), who often manages a set of geo-distributed edge servers [6], for serving their offloading requests. Needless to say, setting an appropriate price is critical to the operator, as an excessively low price will be insufficient to compensate for its operation cost, while an unduly high price will inevitably lead to a decrease in users' computation demands. Previous works mostly use game [9] or auction [10] theory to harmonize the interests of different market participants by setting distinct prices across edge servers. But such pricing schemes are usu-

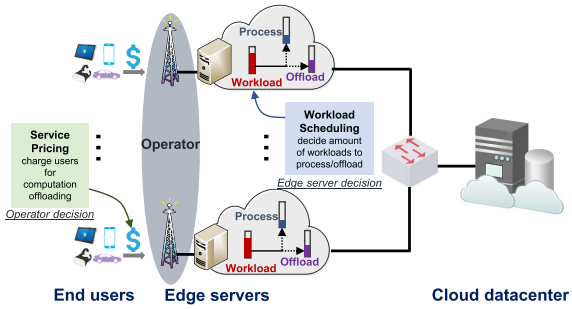


Fig. 1. Illustration of the edge computing system.

ally too complicated for real-world implementation, especially in a highly dynamic environment. A more practical way for the operator is to adaptively select a uniform price for all servers to maximize its utility when providing edge computing services.

There have been many efforts devoted to workload scheduling and service pricing aiming at improving the edge computing performance [11], [12]. However, existing solutions are mostly centralized, which are unscalable to scenarios involving a large number of edge servers [13]. To be more specific, extortionate communication and computation overheads will make it difficult to obtain timely information from many servers in a centralized manner, not to mention determining optimal scheduling and pricing decisions on a global scale. Therefore, one fundamental question is how to *distributedly schedule the workloads and decide the service price to optimize server's scheduling cost and operator's service utility*. To answer this question, researchers are faced with the following challenges.

First, in practice, edge servers dynamically schedule the workloads upon arrival of real-time requests, whereas the price adjustment made by the operator is usually specified for a long period, i.e., daily or weekly basis. Therefore, the former action occurs at a much smaller timescale than the latter one. This brings new modeling requirements for incorporating *interplay and interdependence* between the optimizations of scheduling cost and service utility at different timescales. Second, a *decentralized scheduling scheme* is needed for large-scale edge computing systems. As complete system information is not available, each server has to make its own decisions. However, existing decentralized solutions often have high computation complexity and can not guarantee the optimal performance [14]. One possible remedy is to integrate an estimated global information into the scheduling, but a precise estimation is difficult if the number of servers scales. Third, for easy implementation, the service utility is supposed to be maximized under a *uniform price*. Unfortunately, the state, namely accumulated workload, of geo-distributed edge servers is highly heterogeneous, and hence it is hard to arrive at a uniform price that is optimal for all servers. Considering the price elasticity of computation demand, operator's pricing decision is also intertwined with server's scheduling process, which further complicates the service pricing design.

In this paper, we develop a *decentralized online optimization* for joint workload scheduling and service pricing in a large-scale edge computing system. Specifically, we first propose a *two-timescale optimization framework* to accommodate actions occurring at different timescales, where the scheduling of processing and offloading workloads at a small timescale is embedded into the service pricing at a large timescale. To facilitate decentralized decision making, we harness the theory of *mean field game* to approximate the collective scheduling behavior of all edge servers using a deterministic mean field value [41], [42]. Accordingly, we achieve the minimum scheduling cost for each server only based on its local state. Furthermore, we also leverage the mean field model to characterize the influence of workload scheduling, thus we can transform the dynamic pricing problem into its equivalent form. On this basis, we design a Lyapunov-based pricing scheme to determine the optimal while uniform price over time, which could attain a favorable long-term service utility. In summary, this paper has the following main contributions:

- We propose a two-timescale online optimization framework for joint workload scheduling and service pricing, where a mean field game is employed to facilitate optimal decision making in a *decentralized* manner. To the best of our knowledge, this is the first work that conducts a thorough analysis on the decentralized *online optimal scheduling* for large-scale edge computing systems.
- We devise a mean field model to approximate workload scheduling of all servers via *deterministic ordinary differential equations* (ODEs). Consequently, each server can make the optimal scheduling decisions solely using its local information. We achieve the *closed-form* minimum scheduling cost with $O\left(\frac{1}{\sqrt{M}}\right)$ approximation error, where the error declines as the number of edge servers M increases.
- We transform a complex service utility maximization into a *concise but equivalent* optimization problem by representing heterogeneous server state via its mean field value. Following the Lyapunov optimization approach, we further design an optimal pricing scheme to improve the *long-term utility* while also ensuring load balancing among edge servers, where optimal service price will dynamically *converge* to a steady value when average server workload becomes stable.

II. System Model

We consider a large-scale edge computing system with a set $\mathcal{M} = \{1, 2, \ldots, M\}$ of edge servers which are managed by an operator. Each edge server is endowed with computing functionalities and hence can handle the computation offloading requests sent from end users within its radio coverage. Here each request is typically specified by the task size and required service time for modeling real-world applications (like face recognition and interactive games) [15], [16], i.e., an offloading request corresponds to a certain amount of workloads which could be discrete or continuous values [17]. Upon arrival of user requests, edge servers can process the workloads locally, and also cooperate with the cloud datacenter to further offload computation exceeding their capacity remotely. Along with such request scheduling, a service price is also charged

by the operator when providing computing services for users. Fig. 1 demonstrates a system illustration.

A. Two-Timescale Framework

In most practical scenarios, the price set by an operator often *maintains stable* for a long time, like weekly data plan for end users, while the scheduling of processing and offloading workloads for a server is adapted to real-time user requests, thereby resulting in a two-timescale framework for online workload control. Specifically, edge servers make their processing and offloading decisions in real time, while the service price is dynamically adjusted at a time interval of T duration. Hence, workload scheduling at the small timescale is embedded into service pricing at the large timescale. Formally, we use time slot $n = 1, 2, \ldots$ to denote the large timescale, with each slot lasting for T duration, and time t to denote the small timescale where $t \in [0, T]$ for any time slot n.

B. Server Workload Scheduling

At time t, edge server i receives workloads from users with *stochastic rate* $a_i(t)$, and then needs to make scheduling decisions specified by the local processing speed $s_i(t)$ and offloading speed $o_i(t)$. Let $q_i(t)$ be the workload queue at server i, so that the rate of change for $q_i(t)$ is:

$$dq_i(t) = (a_i(t) - s_i(t) - o_i(t))\, dt. \tag{1}$$

The arriving workloads are usually unbalanced across edge servers, which will deteriorate the system performance if some servers are overloaded. Therefore, load balancing is needed when servers make scheduling decisions [18], [19]. To achieve load balancing, workloads at each server should be balanced around the average level. We define the average queued workload of M edge servers as $\overline{q}^M(t)$, which is:

$$\overline{q}^M(t) = \frac{1}{M} \sum_{i=1}^{M} q_i(t). \tag{2}$$

For edge server i, the scheduling cost $c_i[n]$ in time slot n is:

$$c_i[n] = \mathbb{E}\Big[\int_0^T \big[\alpha s_i^2(t) + \beta o_i^2(t) + \gamma o_i^2(t) + \eta q_i^2(t) + \lambda \left(q_i(t) - \overline{q}^M(t)\right)^2\big] dt\Big]. \tag{3}$$

- $\alpha s_i^2(t)$: local processing cost. It is the energy consumption for processing the workloads, which is proportional to the quadratic processing speed [20].
- $\beta o_i^2(t)$: local offloading cost. Transmission cost of sending workloads to the cloud, which has a quadratic form [21].
- $\gamma o_i^2(t)$: remote processing cost. Cost at the cloud to handle the offloaded computation, and the form is similar to $\alpha s_i^2(t)$.
- $\eta q_i^2(t)$: queue congestion cost. This term represents the congestion level of server i which also implies the processing latency. Convex cost such as the quadratic form is often used [22]. To alleviate congestion, servers have to judiciously decide their processing and offloading speeds.
- $\lambda \left(q_i(t) - \overline{q}^M(t)\right)^2$: load balancing cost. It can cater to unbalanced workloads to facilitate *fair sharing* of cloud resource and *avoiding overloaded situations* since any heavily loaded server will offload more computation [8], [18]. λ is the parameter converting load variance to cost.[1]

When the queue is less congested, the workload sojourn time would decrease. Hence, we can alter the parameter η to reduce the scheduling latency, as discussed in Section IV-E.

The objective for each server i is to minimize the scheduling cost $c_i[n]$ by deciding its processing and offloading speeds $s_i(t), o_i(t)$ based solely on the local state $q_i(t)$. In other words:

$$\begin{aligned} &\min \quad c_i[n] \\ &\text{s.t.} \quad \text{Eq. (1)}, \forall i \in \mathcal{M}. \end{aligned} \tag{4}$$

Due to the dynamics of $q_i(t)$, the average queued workload $\overline{q}^M(t)$ is also dynamic, which in turn influences the action decisions $s_i(t), o_i(t)$. Hence, each scheduling cost $c_i[n]$ is coupled by $\overline{q}^M(t)$, and this coupling increases the difficulty of solving Eq. (4). To handle this issue, we will use a mean field model to decouple the cost minimization problem so as to determine the optimal decisions in a decentralized manner.

C. Operator Service Pricing

Edge servers are managed by an operator, who aims to optimize its service utility. Like pricing scheme adopted by many Internet service providers, a uniform price $p[n]$ is more practical and attractive to end users [23]. In time slot n, the scheduling cost $c_i[n]$ is obtained by solving Eq. (4), and $p[n]$ is the service price set by the operator to charge users for serving their requests. We consider that the average arriving workload $a[n]$ is mainly affected by the charged price to concentrate our attention on pricing scheme design [24], [25]. Concretely, if $p[n]$ increases, users tend to have lower computation demands and submit less requests to edge servers, i.e., lower $a[n]$. W.l.o.g., one could use a function $g(\cdot)$ to relate $p[n]$ and $a[n]$:

$$a[n] = g(p[n]), \tag{5}$$

where $g(\cdot)$ is a decreasing and differentiable function.[2] As for edge server i, the corresponding arrival rate $a_i(t), t \in [0, T]$ in time slot n is a stochastic process characterized by the average arriving workload $a[n]$ [28], [29], that is $\mathbb{E}[a_i(t)] = a[n]$.

Since each time slot n lasts for T duration, the expected total amount of arriving workloads at server i are $\mathbb{E}[\int_0^T a_i(t)dt]$. Hence, the service utility is obtained:

$$u_i[n] = p[n]\mathbb{E}\Big[\int_0^T a_i(t)dt\Big] - c_i[n]. \tag{6}$$

[1]In this paper, we focus on edge-cloud cooperation (offloading) to achieve load balancing, while edge-edge cooperation remains as the future work.

[2]The function $g(\cdot)$ can be obtained by fitting historical data, and is usually time-invariant [26], [27]. Actually, our later pricing scheme design is essentially unchanged even under time-varying $g(\cdot)$. Since we mainly explore the case where offloading demand is influenced by the service price, pricing schemes which also integrate service quality are left as our future work.

Consistent with the queue change in Eq. (1), the queue dynamics from one time slot to the next follows:

$$q_i[n+1] = \max\Big[q_i[n] + \int_0^T a_i(t)dt - \int_0^T (s_i(t) + o_i(t))dt, 0\Big]. \quad (7)$$

The operator's goal is to maximize its long-term average utility via dynamically setting a uniform price $p[n]$ over time:

$$\max \lim_{D\to\infty} \frac{1}{D}\sum_{n=1}^{D}\frac{1}{M}\sum_{i=1}^{M} u_i[n]$$
$$\text{s.t. } \lim_{D\to\infty} \frac{1}{D}\sum_{n=1}^{D} q_i[n] < \infty, \forall i \in \mathcal{M}. \quad (8)$$

The inequality in Eq. (8) means all queues need to be stable. Since queue $q_i[n]$ is stochastic due to the stochastic arrival rate $a_i(t)$, and the queue states among many edge servers are heterogeneous, so that maximizing the utility is difficult through a simple price $p[n]$. We will use the mean field theory to represent $q_i[n]$ by a deterministic mean field value, and this representation is exact when $M \to \infty$. Therefore, we can transform the utility maximization of Eq. (8) into its equivalent problem, which will help design the optimal pricing scheme.

D. Model Discussion

The actions in our model include the processing and offloading workloads by each edge server, and the pricing scheme designed by the operator. Considering the price elasticity of computation demand, each server decides its optimal processing and offloading speeds at the small timescale to minimize the scheduling cost, since the service price remains unchanged during any time slot. Through the mean field analysis, we can attain minimum scheduling cost in a decentralized manner. By deriving an optimal price at the large timescale, the operator achieves to maximize its long-term service utility where the scheduling cost is also included in the utility calculation. Though this intertwined modeling raises difficulties of optimizing server's scheduling cost and operator's service utility, it more conforms to real-world situations and also enables a *finer-grained workload control* compared to previous works, especially for large-scale edge computing. In general, our proposed two time-scale framework is also applicable to other scheduling or pricing applications when involving coupled fast and slow procedures. Besides, the characterized mean field model can further provide a new insight for designing optimal decentralized scheduling schemes in many other related areas like cellular networks, data centers, etc. The main notations in this paper are listed in Table I.

TABLE I
NOTATION

Notation	Description
t, n	time index, time slot index
T, M	length of small timescale, number of servers
$a_i(t), a[n]$	arrival rate, average arriving workload
$s_i(t), s(t)$	processing speed of server i, a server
$o_i(t), o(t)$	offloading speed of server i, a server
$q_i(t)/q_i[n], q(t)/q[n]$	workload queue at server i, a server
$\overline{q}^M(t), \overline{q}(t)/\overline{q}[n]$	average, expected queued workload
$c_i(t)/c_i[n], c(t)$	scheduling cost of server i, a server
$p[n]$	service price in time slot n
$u_i[n], \overline{u}[n]$	utility of server i, in mean field model
σ_i^2, σ^2	randomness magnitude to server i, a server
W_i, W	Brownian motion to server i, a server
$x(t), y(t), z(t)$	cost coefficients in mean field model
$s_i^M(t)$	processing speed in M-edge server system
$o_i^M(t)$	offloading speed in M-edge server system
$x^M(t), y^M(t)$	cost coefficients in M-edge server system
$c_i^M[n]$	scheduling cost in M-edge server system
$g(\cdot), h(\cdot)$	price-workload function, reverse function
V	importance weight
$f(\cdot)$	drift-minus-utility function

III. Mean Field Based Decentralized Scheduling

In this section, we aim to derive the minimum scheduling cost. To this end, one has to solve Eq. (4) for M edge servers due to the coupling by the average queued workload $\overline{q}^M(t)$. It is difficult to achieve this goal because of two reasons. First, simultaneously solving M-coupled optimization problems is computationally prohibitive when M is very large. Second, the dynamics of queue $q_i(t)$ will lead to a rapidly changing $\overline{q}^M(t)$, which makes it hard to track this coupling term. Hence, we use the mean field model to consider large value of M. Specifically, we leverage a deterministic value $\overline{q}(t)$ to approximate $\overline{q}^M(t)$ in the mean field model. Therefore, we can *decouple* the M cost minimization problems by regarding $\overline{q}(t)$ as an *exogenous parameter* since an individual edge server has *infinitesimal influence* on $\overline{q}(t)$ [42]. Through this mean field approximation, we are able to deal with each cost minimization individually.

A. Mean Field Approximation

The scheduling cost of server i is related to the stochastic arrival rate $a_i(t)$ which influences the workload queue in Eq. (1). As $a_i(t)$ has a mean value of average arriving workload $a[n]$, a widely adopted model is that $a_i(t)$ is a random value following a Gaussian distribution $\mathcal{N}(a[n], \sigma_i^2)$ [28], where each $\mathcal{N}(a[n], \sigma_i^2)$ is assumed to be independent [41], [42] and σ_i^2 is the randomness magnitude specified to server i. Also denote $\mathbb{E}[\sigma_i^2] = \overline{\sigma}^2$ as the expected randomness magnitude of the edge computing system. In general, the Gaussian model $a_i(t) \sim \mathcal{N}(a[n], \sigma_i^2)$ can be extended to $a_i(t) \sim \mathcal{N}(\overline{a}(t), \sigma_i^2)$, with $\overline{a}(t)$ varying over time t and $\frac{1}{T}\int_0^T \overline{a}(t)dt = a[n]$ in time slot n. These two models share the same spirit pertaining to later mean field approximation and scheduling cost derivation, so that we utilize the concise one, i.e., $a_i(t) \sim \mathcal{N}(a[n], \sigma_i^2)$, for ease of exposition.

Let the cost $c_i(t), t \in [0, T]$ be:

$$c_i(t) = \mathbb{E}\Big[\int_t^T \big[\alpha s_i^2(\tau) + \beta o_i^2(\tau) + \gamma o_i^2(\tau) + \eta q_i^2(\tau) + \lambda\left(q_i(\tau) - \overline{q}^M(\tau)\right)^2\big]d\tau\Big]. \quad (9)$$

In time slot n, $c_i[n]$ in Eq. (3) is actually $c_i(0)$ in Eq. (9), i.e., lower limit $t = 0$. Hereinafter, we remove the subscript i in parameters (workload queue, action, cost, etc.) for brevity.

Replace $\overline{q}^M(t)$ with $\overline{q}(t)$, then $c(t) = \mathbb{E}\big[\int_t^T [\alpha s^2(\tau) + \beta o^2(\tau) + \gamma o^2(\tau) + \eta q^2(\tau) + \lambda(q(\tau) - \overline{q}(\tau))^2] d\tau\big]$. Now our objective is to minimize $c(t)$ using $\overline{q}(t)$. In line with the queue dynamics of Eq. (1), we have:

$$dq(t) = (a[n] - s(t) - o(t))dt + \sigma dW, \tag{10}$$

where W is the Brownian motion since $a(t)dt = a[n]dt + \sigma dW$ when $a(t) \sim \mathcal{N}(a[n], \sigma^2)$. Basically, dW, or dW_i if adding the subscript i, denotes the randomness from stochastic arrival rate $a(t)$, or $a_i(t)$. Akin to [29], [42], and [44], we mainly use Brownian motion to model the workload randomness, and will also ignore the case of negative values, which indeed rarely occurs as shown in later performance evaluations.

1) Optimal Actions: The optimal actions, i.e., the processing and offloading speeds $s(t), o(t)$, are attained based on the Hamilton–Jacobi–Bellman equation [30]. Mathematically, we solve the following partial differential equation to derive the optimal $s(t), o(t)$ and obtain the minimum cost $c(t)$:

$$\begin{aligned} -\frac{\partial c}{\partial t} = \min_{s(t),o(t)} & \alpha s^2(t) + \beta o^2(t) + \gamma o^2(t) + \eta q^2(t) \\ & + \lambda(q(t) - \overline{q}(t))^2 + (a[n] - s(t) - o(t))\frac{\partial c}{\partial q} \\ & + \frac{1}{2}\sigma^2 \frac{\partial^2 c}{\partial q^2}. \end{aligned} \tag{11}$$

Since the right-hand side is a quadratic polynomial, the optimal processing and offloading speeds are computed as:

$$\begin{aligned} s(t) &= \frac{1}{2\alpha}\frac{\partial c}{\partial q}, \\ o(t) &= \frac{1}{2(\beta+\gamma)}\frac{\partial c}{\partial q}. \end{aligned} \tag{12}$$

In view of the cost function and the queue dynamics, the cost minimization is a linear-quadratic-Gaussian (LQG) control problem. As LQG has a quadratic solution [30], a feasible way to deduce the solution is to first construct a quadratic formula and then solve the corresponding coefficients of the constructed formula [42]. Regarding the minimum cost $c(t)$, we consider it has the following expression:

$$c(t) = x(t)q^2(t) + y(t)q(t) + z(t), \tag{13}$$

where $x(t), y(t), z(t)$ are time-varying coefficients to be solved. Also, it results in $\frac{\partial c}{\partial t} = \frac{dx}{dt}q^2(t) + \frac{dy}{dt}q(t) + \frac{dz}{dt}$, $\frac{\partial c}{\partial q} = 2x(t)q(t) + y(t)$, and $\frac{\partial^2 c}{\partial q^2} = 2x(t)$.

From Eq. (13), we rewrite the optimal actions in Eq. (12):

$$\begin{aligned} s(t) &= \frac{2x(t)q(t) + y(t)}{2\alpha}, \\ o(t) &= \frac{2x(t)q(t) + y(t)}{2(\beta+\gamma)}. \end{aligned} \tag{14}$$

Because $2x(t)q(t) + y(t) \geq 0$, then $s(t) \geq 0$ and $o(t) \geq 0$. Moreover, the optimal processing and offloading speeds $s(t), o(t)$ are deterministic in terms of the queue $q(t)$ even though the arrival rate $a(t)$ is stochastic, as we obtain $s(t), o(t)$ by minimizing the cost $c(t)$, an expected value, in Eq. (9).

2) Mean Field Model: Now we elucidate the decoupling of the M cost minimization problems in the mean field model. Substituting Eq. (14) into Eq. (11), we obtain:

$$\begin{aligned} & -\frac{dx}{dt}q^2(t) - \frac{dy}{dt}q(t) - \frac{dz}{dt} \\ &= -\left[\frac{1}{4\alpha} + \frac{1}{4(\beta+\gamma)}\right](2x(t)q(t) + y(t))^2 \\ &\quad + \eta q^2(t) + \lambda(q(t) - \overline{q}(t))^2 \\ &\quad + a[n](2x(t)q(t) + y(t)) + \sigma^2 x(t). \end{aligned} \tag{15}$$

Since Eq. (15) holds for any $q(t)$, the coefficients on both sides should be consistent. Let $b = \frac{1}{\alpha} + \frac{1}{\beta+\gamma}$, and it yields:

$$\frac{dx}{dt} = bx^2(t) - (\eta + \lambda), \tag{16}$$

$$\frac{dy}{dt} = bx(t)y(t) - 2a[n]x(t) + 2\lambda\overline{q}(t), \tag{17}$$

$$\frac{dz}{dt} = -\sigma^2 x(t) + \frac{b}{4}y^2(t) - a[n]y(t) - \lambda\overline{q}^2(t). \tag{18}$$

Note that $\overline{q}(t)$ is an exogenous term in Eqs. (17) and (18) because the dynamics of a queue $q(t)$ will not influence $\overline{q}(t)$ when $M \to \infty$. Hence, each server can decide its optimal actions based on this exogenous $\overline{q}(t)$ in the mean field model, rather than the coupling term $\overline{q}^M(t)$. In other words, each queue $q(t)$ is asymptotically independent and the original cost minimization problems are decoupled.

Considering the deterministic value $\overline{q}(t)$ is an approximation of the empirical average queued workload $\overline{q}^M(t)$, we regard $\overline{q}(t)$ as the *expected queued workload* $\mathbb{E}[q(t)]$. Since each stochastic queue $q(t)$ has a bounded variance, we claim that $\overline{q}^M(t)$ will converge to $\overline{q}(t)$ when $M \to \infty$ based on the asymptotic independence analysis, which will be formally presented in Theorem 1. To characterize $\overline{q}(t)$, we take expectation of the queue dynamics $dq(t) = (a[n] - s(t) - o(t))dt + \sigma dW$ in Eq. (10), and integrate optimal actions in Eq. (14). We have:

$$\frac{d\overline{q}}{dt} = a[n] - \overline{s}(t) - \overline{o}(t) = -bx(t)\overline{q}(t) - \frac{1}{2}by(t) + a[n]. \tag{19}$$

The randomness in stochastic arrival rate $a(t)dt = a[n]dt + \sigma dW$ is removed due to $\mathbb{E}[\sigma dW] = 0$. Eq. (19) reveals that one can track the deterministic $\overline{q}(t)$ in a *decentralized manner by solving the ODE*. Therefore, the mean field model corresponds to the *deterministic ODE system* of Eqs. (16)-(19).

In order to derive the minimum cost $c(t)$ and optimal actions $s(t), o(t)$, we can solve the ODE system distributedly as a result of the decoupling via the mean field approximation. From Eq. (9), we know that $c(T) = 0$. Because the minimum cost has a quadratic expression of Eq. (13), the terminal conditions for $x(t), y(t), z(t)$ satisfy:

$$x(T) = y(T) = z(T) = 0. \tag{20}$$

For $\overline{q}(t)$, its initial value and derivation at $t = T$ are obtained when we combine with Eq. (19) and Eq. (20):

$$\begin{aligned} \overline{q}(0) &= \mathbb{E}[q[n]] = \overline{q}[n], \\ \frac{d\overline{q}}{dt}\Big|_{t=T} &= a[n]. \end{aligned} \tag{21}$$

B. Closed-Form Quadratic Cost

We need to calculate coefficients $x(t), y(t), z(t)$ in Eq. (13) to derive the minimum cost $c(t)$. To this end, we do the following. First, we solve the ODE of $x(t)$ as Eq. (16) does not depend on other variables. Second, we derive the expressions for $y(t)$ and $\overline{q}(t)$. This is because Eqs. (17) and (19), $y(t)$ and $\overline{q}(t)$, are inter-dependent provided the result of $x(t)$. Third, we determine the term $z(t)$ since the ODE Eq. (18) involves coefficients $x(t), y(t)$ and the value $\overline{q}(t)$. Using the conditions in Eqs. (20) and (21), we summarize our results as follows (see Appendix A in supplementary material for the proof):

Lemma 1: The expression for $x(t), t \in [0, T]$ satisfies:

$$x(t) = \sqrt{\frac{\eta+\lambda}{b}} \frac{1 - e^{2\sqrt{(\eta+\lambda)b}(t-T)}}{1 + e^{2\sqrt{(\eta+\lambda)b}(t-T)}}. \tag{22}$$

Lemma 2: The expressions for $\overline{q}(t), y(t), t \in [0, T]$ are:

$$\overline{q}(t) = A_1 e^{\sqrt{\eta b}t} + A_2 e^{-\sqrt{\eta b}t}, \tag{23}$$

$$y(t) = -2x(t)\overline{q}(t) - 2\sqrt{\frac{\eta}{b}}\left(A_1 e^{\sqrt{\eta b}t} - A_2 e^{-\sqrt{\eta b}t}\right) + \frac{2a[n]}{b}, \tag{24}$$

where $x(t)$ is from Eq. (22) *and*

$$A_1 = \frac{\overline{q}[n]\sqrt{\eta b}e^{-\sqrt{\eta b}T} + a[n]}{\sqrt{\eta b}e^{\sqrt{\eta b}T} + \sqrt{\eta b}e^{-\sqrt{\eta b}T}},$$
$$A_2 = \frac{\overline{q}[n]\sqrt{\eta b}e^{\sqrt{\eta b}T} - a[n]}{\sqrt{\eta b}e^{\sqrt{\eta b}T} + \sqrt{\eta b}e^{-\sqrt{\eta b}T}}. \tag{25}$$

Lemma 3: The expression for $z(t), t \in [0, T]$ is:

$$z(t) = x(t)\overline{q}^2(t) + \frac{\sigma^2}{b}\ln\left[1 + e^{2\sqrt{(\eta+\lambda)b}(t-T)}\right] - \sigma^2\sqrt{\frac{\eta+\lambda}{b}}t + \sqrt{\frac{\eta}{b}}\left(A_1^2 e^{2\sqrt{\eta b}t} - A_2^2 e^{-2\sqrt{\eta b}t}\right) - \frac{a^2[n]}{b}t + A_3, \tag{26}$$

where $x(t), \overline{q}(t)$ are from Eqs. (22)-(23)*, A_1, A_2 are defined in Eq.* (25) *and A_3 is*:

$$A_3 = -\frac{2\overline{q}[n]a[n]}{b\left(e^{\sqrt{\eta b}T} + e^{-\sqrt{\eta b}T}\right)} - \frac{e^{\sqrt{\eta b}T} - e^{-\sqrt{\eta b}T}}{e^{\sqrt{\eta b}T} + e^{-\sqrt{\eta b}T}} \frac{a^2[n]}{b\sqrt{\eta b}} + \frac{a^2[n]}{b}T + \sigma^2\sqrt{\frac{\eta+\lambda}{b}}T - \frac{\sigma^2}{b}\ln 2. \tag{27}$$

One of our main contributions is deriving the *closed-form minimum cost* $c(t)$ for any $q(t)$ according to Lemmas 1-3, which is not available in previous mean field works [29], [42], [44]. This enables a more subtle characterization of the workload scheduling when compared with mere numerical results, and also provides new insights for other mean field based scheme design. From Eqs. (9) and (13), the minimum scheduling cost $c[n]$ in time slot n turns out to be:

$$c[n] = c(0) = x(0)q^2(0) + y(0)q(0) + z(0). \tag{28}$$

Specifically, the cost $c[n]$ is $c_i[n]$ of edge server i when we let the parameters $q(0), \sigma$ be $q_i(0), \sigma_i$, respectively. More importantly, this explicit cost expression allows us to compute and to maximize the service utility as we will show in Section IV, instead of only relying on numerical methods which are not intuitive and may introduce undesired errors.

Remark: Previous results are attained provided that the arriving workload $a_i(t) \sim \mathcal{N}(a[n], \sigma_i^2)$ has a uniform mean value $a[n]$, which is commonly adopted in [31] and [41]. Though simple, this model is typical and can guide the design of optimal workload scheduling in more sophisticated scenarios. For instance, consider that $a_i(t) \sim \mathcal{N}(a_i[n], \sigma_i^2)$ with different $a_i[n]$ across different servers. Denote $\overline{a}[n] = \mathbb{E}[a[n]]$, $\overline{x}(t) = \mathbb{E}[x(t)]$ and $\overline{y}(t) = \mathbb{E}[y(t)]$ if removing the subscript i. Then Eq. (19) will change to $\frac{d\overline{q}}{dt} = -b\overline{x}(t)\overline{q}(t) - \frac{1}{2}b\overline{y}(t) + \overline{a}[n]$ and Eqs. (16)-(18) are specified for each server accordingly, i.e, the mean field model is still effective. Following a similar approach, $\overline{x}(t) = x(t)$ and $\overline{q}(t)$ will be the same as Eq. (22) and Eq. (23), respectively. However, each $y(t)$ and $z(t)$ can only be obtained numerically. As a result, one can not characterize the explicit cost $c(t)$ regarding to $\overline{a}[n]$, not to mention devising the optimal pricing scheme for the operator. Nonetheless, our evaluations in Section V show that costs of uniform and heterogeneous arriving workloads have a similar shape, that is the uniform case with closed-form expressions can give us insights for more complicated scheduling design.

C. Approximation Error Analysis

Before analyzing the approximation error of the mean field model, we introduce the concept of ϵ-equilibrium [42].

Definition 1: Let $\pi_i = \{s_i, o_i\}$ be the strategy of edge server i where s_i, o_i are the processing and offloading speeds. Denote $\pi_{-i} = \{s_{-i}, o_{-i}\}$ as the strategies of edge servers except i. $\pi^* = \{\pi_i^*, \pi_{-i}^*\}$ is called an equilibrium strategy, if the scheduling cost satisfies:

$$c_i(\pi^*) \leq c_i\left(\{\pi_i, \pi_{-i}^*\}\right), \forall i \in \mathcal{M}.$$

A strategy π^ϵ is an ϵ-equilibrium if there is $\epsilon \geq 0$ such that:

$$c_i(\pi^\epsilon) \leq c_i\left(\{\pi_i, \pi_{-i}^\epsilon\}\right) + \epsilon, \forall i \in \mathcal{M}.$$

In Definition 1, an ϵ-equilibrium is actually an equilibrium when $\epsilon = 0$. Later on, we will demonstrate that the optimal actions derived in the mean field model is an ϵ-equilibrium, and $\epsilon \to 0$ if $M \to \infty$. Based on the mean field approximation, the deterministic value $\overline{q}(t)$ is used to represent the average queued workload $\overline{q}^M(t)$, so that the closed-from cost could be obtained. Therefore, we need to characterize the accuracy of this representation. To avoid confusion, we will use superscript M to denote values acquired through $\overline{q}^M(t)$.

1) Approximation Error of Queue Size: In the original M-edge server system, the optimal processing and offloading speeds are $s_i^M(t)$ and $o_i^M(t)$, which have the expressions when following the same approach of deriving Eq. (14) as:

$$s_i^M(t) = \frac{2x^M(t)q_i(t) + y^M(t)}{2\alpha},$$
$$o_i^M(t) = \frac{2x^M(t)q_i(t) + y^M(t)}{2(\beta+\gamma)}, \tag{29}$$

where $x^M(t), y^M(t)$ are the corresponding cost coefficients similar to Eq. (13). We reiterate that all parameters, like workload queue, action, coefficient, cost, etc., with superscript M are obtained from using the average queued workload $\overline{q}^M(t)$ in the M-edge server system. Besides,

one can not derive explicit expressions for those parameters due to the coupled M cost minimization problems. According to Eq. (29), the queue dynamics in Eq. (1) becomes $dq_i(t) = \big(a[n] - s_i^M(t) - o_i^M(t)\big)\,dt + \sigma_i dW_i = \big(a[n] - bx^M(t)q_i(t) - \frac{1}{2}by^M(t)\big)\,dt + \sigma_i dW_i$ with $b = \frac{1}{\alpha} + \frac{1}{\beta+\gamma}$ since $a_i(t) \sim \mathcal{N}(a[n], \sigma_i^2)$. Considering the optimal actions in Eq. (14), we can also define $dv_i(t) = \big(a[n] - bx(t)v_i(t) - \frac{1}{2}by(t)\big)\,dt + \sigma_i dW_i$ as the queue dynamics derived from $\overline{q}(t)$ in the mean field model. The following theorem states the approximation error of the queue size.

Theorem 1: The error between $\overline{q}^M(t), q_i(t)$ and $\overline{q}(t), v_i(t)$ satisfies: $\sup_{t\in[0,T]} \mathbb{E}[|\overline{q}^M(t) - \overline{q}(t)| + |q_i(t) - v_i(t)|] < \epsilon', \forall i \in \mathcal{M}$, *where* $\epsilon' = O\left(\frac{1}{\sqrt{M}}\right)$.

Please refer to Appendix B-A in supplementary material for the proof. This theorem indicates that the error between the queue size in the mean field model and that in the M-edge server system is $O\left(\frac{1}{\sqrt{M}}\right)$, i.e., the system will converge to the mean field limit characterized by Eqs. (16)-(19).

2) ϵ-Equilibrium: Based on Theorem 1, we now prove the ϵ-equilibrium of the mean field model. The scheduling cost in Eq. (3) obtained from $\overline{q}^M(t)$ is $c_i^M[n] = \mathbb{E}\big[\int_0^T [\alpha(s_i^M(t))^2 + \beta(o_i^M(t))^2 + \gamma(o_i^M(t))^2 + \eta q_i^2(t) + \lambda(q_i(t) - \overline{q}^M(t))^2]dt\big]$ with $s_i^M(t), o_i^M(t)$ in Eq. (29). For the cost in the mean field model, we have $c_i[n] = \mathbb{E}\big[\int_0^T [\alpha s_i^2(t) + \beta o_i^2(t) + \gamma o_i^2(t) + \eta v_i^2(t) + \lambda(v_i(t) - \overline{q}(t))^2]dt\big]$ where $s_i(t), o_i(t)$ are from Eq. (14). One can expand each term of the two costs, and this shows optimal actions $s_i(t), o_i(t)$ computed from $\overline{q}(t)$ satisfy ϵ-equilibrium.

Theorem 2: The mean field model satisfies ϵ-equilibrium: $\big|c_i^M[n] - c_i[n]\big| \le \epsilon$, *with* $\epsilon = O\left(\frac{1}{\sqrt{M}}\right)$.

See Appendix B-B in supplementary material for the proof. Theorem 2 implies that the minimum cost, incurred by the optimal processing and offloading speeds deduced from the mean field approximation, has a small deviation to the minimum cost for the M-edge server system. Therefore, we can compute the actions and cost in a decentralized manner in the mean field model, where both queue and cost approximation errors are only $O\left(\frac{1}{\sqrt{M}}\right)$. Note that $\epsilon \to 0$ when $M \to \infty$, and the ϵ-equilibrium will eventually converge to an exact equilibrium.

IV. Dynamic Service Pricing Scheme

In the previous section, the minimum cost of workload scheduling is obtained. Given this cost, we proceed to design a pricing scheme for the operator to maximize its service utility. As a uniform price is more preferred in practice [23], the pricing scheme should determine a price which stays fixed during each time slot for all servers. Since the queue dynamics in Eq. (7) is stochastic, the queue state of each edge server is heterogeneous, so that we need to simplify the complex utility maximization problem of Eq. (8) into a concise form, which will only entail a deterministic queue value.

A. Transformation of Utility Maximization

We first calculate the utility using the closed-form cost. Let $q(0) = q_i[n], \sigma = \sigma_i$ in Eq. (28), and the cost of edge server i is $c_i[n] = x(0)q_i^2[n] + y(0)q_i[n] + z_i(0)$ with only $z_i(0)$ related to σ_i^2 according to Lemma 3. From Eq. (6), the utility is:

$$\begin{aligned} u_i[n] &= p[n]\mathbb{E}\Big[\int_0^T a_i(t)dt\Big] - x(0)q_i^2[n] - y(0)q_i[n] - z_i(0) \\ &= Tp[n]a[n] - x(0)q_i^2[n] - y(0)q_i[n] - z_i(0), \end{aligned} \tag{30}$$

where the second equality is because $a_i(t) \sim \mathcal{N}(a[n], \sigma_i^2)$ and $\mathbb{E}[\int_0^T a_i(t)dt] = Ta[n]$. The queue $q_i[n]$ is *stochastic and heterogeneous* for different server i due to its random dynamics, and this makes it difficult to directly optimize the long-term average utility in Eq. (8). Therefore, we transform the utility maximization into a succinct problem which harnesses the deterministic queue $\overline{q}[n]$ to represent the stochastic $q_i[n]$. To simplify notations, we denote m_1, m_2 as:

$$\begin{aligned} m_1 &= \frac{e^{\sqrt{\eta b}T} - e^{-\sqrt{\eta b}T}}{\sqrt{\eta b}e^{\sqrt{\eta b}T} + \sqrt{\eta b}e^{-\sqrt{\eta b}T}}, \\ m_2 &= \frac{1}{e^{\sqrt{\eta b}T} + e^{-\sqrt{\eta b}T}}. \end{aligned} \tag{31}$$

1) Deterministic Service Utility: From Eq. (23), the queue $\overline{q}(t)$ in the mean field model is deterministic, and satisfies: $\overline{q}[n] = \overline{q}(0)$, $\overline{q}[n+1] = \max[\overline{q}(T), 0] = \overline{q}(T)$ in time slot n. According to Eq. (25) and using m_1, m_2 in Eq. (31), we have a simple form:

$$\overline{q}[n+1] = m_1 a[n] + 2m_2\overline{q}[n]. \tag{32}$$

Define a service utility $\overline{u}[n] \triangleq Tp[n]a[n] - x(0)\overline{q}^2[n] - y(0)\overline{q}[n] - z(0)$, where $\sigma^2 = \overline{\sigma}^2$ for $z(0)$. Now we transform the average utility into this deterministic service utility $\overline{u}[n]$.

Theorem 3: The utility maximization of Eq. (8) *is equivalent to the following problem in the mean field limit, or $M \to \infty$:*

$$\begin{aligned} &\max \lim_{D\to\infty} \frac{1}{D}\sum_{n=1}^{D} \overline{u}[n] \\ &\text{s.t. } \lim_{D\to\infty} \frac{1}{D}\sum_{n=1}^{D} \overline{q}[n] < \infty. \end{aligned} \tag{33}$$

See Appendix C-A in supplementary material for the detailed proof. Theorem 3 enables designing the pricing scheme based on a concise yet equivalent problem. For the service utility $\overline{u}[n]$, we express it in terms of the average arriving workload $a[n]$ below.

Proposition 1: The expression for the service utility $\overline{u}[n]$ is:

$$\overline{u}[n] = Tp[n]a[n] - \overline{x}a^2[n] - \overline{y}a[n] - \overline{z}, \tag{34}$$

where

$$\begin{aligned} \overline{x} &= \frac{T - m_1}{b},\ \overline{y} = \frac{2\overline{q}[n]}{b}(1 - 2m_2), \\ \overline{z} &= \eta m_1 \overline{q}^2[n] + \frac{\overline{\sigma}^2}{b}\ln\left(1 + e^{-2\sqrt{(\eta+\lambda)b}T}\right) \\ &\quad + \overline{\sigma}^2\sqrt{\frac{\eta+\lambda}{b}}T - \frac{\overline{\sigma}^2}{b}\ln 2. \end{aligned} \tag{35}$$

Please refer to Appendix C-B in supplementary material for the proof. On the basis of Eq. (5), the average arriving workload (workload for brevity) is related to the service price set by the operator, i.e., $a[n] = g(p[n])$ where $g(\cdot)$ is a

decreasing and differentiable function. Because the utility $\overline{u}[n]$ is formulated pertaining to $a[n]$, we utilize the inverse function $p[n] = g^{-1}(a[n]) \triangleq h(a[n])$ to calculate $\overline{u}[n]$, and $h(a[n])$ has the same monotonicity property as $g(p[n])$. Hence, the service utility in Eq. (34) becomes $\overline{u}[n] = Th(a[n])a[n] - \overline{x}a^2[n] - \overline{y}a[n] - \overline{z}$. Originally, the operator will design a dynamic pricing scheme to maximize its long-term utility. Since it is more convenient to express $\overline{u}[n]$ via the workload $a[n]$ which is altered by the price $p[n]$, deriving the optimal price can be considered as *computing the optimal workload*. In the following, we regard $a[n]$ as the controlled action, and it has the same effect as setting a uniform price $p[n]$ due to the mapping $p[n] = h(a[n])$.

2) Characteristics of Transformed Problem: To optimize the long-term utility, we actually face a dynamic programming problem in that the utility $\overline{u}[n]$ depends on both the action $a[n]$ and the queue (state) $\overline{q}[n]$. However, $\overline{u}[n]$ is non-positive definite, and $a[n] \neq 0$ given $\overline{q}[n] = 0$. It is impossible to attain the theoretically optimal utility even by adaptive dynamic programming like value iteration [32] and policy iteration [33], because they are computationally expensive and may not converge due to the characteristics of $\overline{u}[n]$. To obtain a larger long-term utility, we employ Lyapunov optimization.

B. Lyapunov Optimization

According to Proposition 1, we observe that if the queue $\overline{q}[n]$ increases, then the utility $\overline{u}[n]$ will decrease. Therefore, to optimize the long-term utility, one should both *minimize the queue size and maximize the immediate utility simultaneously*, which can be achieved by Lyapunov optimization.

1) Lyapunov Drift: We utilize $\overline{q}[n]$ as the Lyapunov function, and define the Lyapunov drift as the change of the Lyapunov function from one time slot to the next:

$$\Delta(\overline{q}[n]) = \overline{q}[n+1] - \overline{q}[n]. \tag{36}$$

Note that we do not apply the quadratic Lyapunov function $\frac{1}{2}\overline{q}^2[n]$, as used in many previous works [34], [35]. This is because we can directly calculate the Lyapunov drift, while previous works aim to derive a bound of the drift aided by the quadratic form. Based on the Lyapunov drift theorem (Theorem 4.1 in [35]), one can drive the queue $\overline{q}[n]$ to a small value by greedily minimizing the Lyapunov drift $\Delta(\overline{q}[n])$. From Eq. (32), we obtain $\Delta(\overline{q}[n])$ as:

$$\Delta(\overline{q}[n]) = m_1 a[n] + 2m_2\overline{q}[n] - \overline{q}[n]. \tag{37}$$

2) Drift-Minus-Utility: To coordinate queue minimization and utility maximization, we calculate the drift-minus-utility:

$$\begin{aligned} f(a[n]) = \Delta(\overline{q}[n]) - V\overline{u}[n] = V\overline{x}a^2[n] + (V\overline{y} + m_1)a[n] \\ -VTh(a[n])a[n] + V\overline{z} + 2m_2\overline{q}[n] - \overline{q}[n], \end{aligned} \tag{38}$$

where V is the importance weight on the utility term, m_1, m_2 are from Eq. (31) and $\overline{x}, \overline{y}, \overline{z}$ are given in Eq. (35). In line with the Lyapunov optimization theorem (Theorem 4.2 in [35]), greedily minimizing the drift-minus-utility $f(a[n])$ can strike a tradeoff between the queue minimization and the utility maximization. Therefore, one needs to derive the optimal price $p^*[n]$ to minimize $f(a[n])$, which is converted to calculating the optimal workload $a^*[n]$. Next, we first assume the optimal workload, $a^*[n] = \arg\min_{a[n]} f(a[n])$, is already derived to present the results, and then demonstrate how to compute this $a^*[n]$ in different circumstances.

3) Queue Stability: The utility maximization of Eq. (8) is subject to the constraint that each queue $q_i[n]$ is stable, which is transformed into the stability constraint of the deterministic queue $\overline{q}[n]$ in Theorem 3. To show that $\overline{q}[n]$ is stable, we introduce an auxiliary lemma to bound the optimal workload. See Appendix D-A in supplementary material for the proof.

Lemma 4: From Eq. (35)*,* $\overline{x} > 0$ *and* $\overline{y} \geq 0$*. There exists a finite upper bound* C^* *for the optimal workload* $a^*[n]$*.*

The following theorem states the queue stability.

Theorem 4: The queue $\overline{q}[n]$ *is stable and satisfies*:

$$\lim_{D\to\infty} \frac{1}{D}\sum\nolimits_{n=1}^{D} \overline{q}[n] \leq \frac{m_1}{1-2m_2}C^*, \tag{39}$$

where C^* *is the upper bound in Lemma 4.*

See Appendix D-B in supplementary material for the proof. Theorem 4 implies that no edge servers will be overloaded, and hence load balancing is achieved. In [34], [35], one could simultaneously maintain the queue stability, and obtain a long-term utility which has $O(1/V)$ gap to the theoretically optimal value by minimizing the drift-minus-utility with V being the importance weight. The $O(1/V)$ asymptotical utility can not be derived in this paper as $\overline{u}[n]$ depends on both the action $a[n]$ and the state $\overline{q}[n]$, while the utility in [34], [35] is determined only by the actions. However, minimizing the drift-minus-utility still ensures a larger long-term utility. By doing so, one can avoid myopically maximizing the one-slot utility, which is potentially detrimental to improving the long-term utility. We will explicitly show this point in the performance evaluation.

C. Optimal Workload

Remember that finding the optimal price is converted to finding the optimal workload since the workload is altered by the service price. According to Eq. (38), we take the first and second order derivatives of the drift-minus-utility $f(a[n])$ over the workload $a[n]$:

$$\begin{aligned} \frac{df}{da} &= 2V\overline{x}a[n] + V\overline{y} + m_1 - VTh(a[n]) - VTa[n]\frac{dh}{da}, \\ \frac{d^2f}{da^2} &= 2V\overline{x} - 2VT\frac{dh}{da} - VTa[n]\frac{d^2h}{da^2}. \end{aligned} \tag{40}$$

There will exist only one minimizer if $f(a[n])$ is convex, or $\frac{d^2f}{da^2} > 0$. From Lemma 4, we know that $\overline{x} > 0$, so the convexity condition is relaxed to:

$$2\frac{dh}{da} + a[n]\frac{d^2h}{da^2} \leq 0. \tag{41}$$

Considering that both the workload $a[n]$ and the price $p[n]$ are non-negative, we combine with exact forms of the decreasing reverse function $h(a[n])$ to obtain the optimal workload $a^*[n]$ as discussed below, where all coefficients are positive.

1) Linear Function: The first example is $h(a[n]) = -C_p a[n] + C_{\max}$. It can be easily verified that $f(a[n])$ is convex because the condition in Eq. (41) holds. Let $\frac{df}{da} = 0$, we have $a^*[n] = \frac{VTC_{\max} - (V\overline{y}+m_1)}{2V\overline{x}+2VTC_p}$.

2) Quadratic Function: Assume $h(a[n]) = -C_{p1}a^2[n] - C_{p2}a[n] + C_{\max}$, and Eq. (41) is still satisfied. The optimal workload is derived by setting $\frac{df}{da} = 0$, and we obtain $a^*[n] = -\frac{V\overline{x}+VTC_{p2}}{3VTC_{p1}} + \frac{\sqrt{(V\overline{x}+VTC_{p2})^2+3VTC_{p1}(VTC_{\max}-V\overline{y}-m_1)}}{3VTC_{p1}}$.

3) Logarithm Function: Suppose $h(a[n]) = \ln\frac{C_{\max}}{1+C_p a[n]}$, and hence the condition in Eq. (41) is satisfied. Let $\frac{df}{da} = 0$, and then we need to numerically solve the equation $2V\overline{x}a^*[n] + V\overline{y} + m_1 - VT\ln\frac{C_{\max}}{1+C_p a^*[n]} + VT\frac{C_p a^*[n]}{1+C_p a^*[n]} = 0$ to derive the optimal workload $a^*[n]$.

4) Exponential Function: Lastly, we consider $h(a[n]) = C_{\max}e^{-C_p a[n]}$. It can be proved that if $\overline{x} \geq \frac{1}{2}TC_{\max}C_p e^{-3}$, then $f(a[n])$ is convex. Otherwise, one may need to compare $f(a[n])$ at different extreme points. Using $\frac{df}{da} = 0$, we obtain $a^*[n]$ by numerically solving the equation $2V\overline{x}a^*[n] + V\overline{y} + m_1 - VTC_{\max}(e^{-C_p a^*[n]} - C_p e^{-C_p a^*[n]}a^*[n]) = 0$.

D. Optimal Workload Analysis

In the previous subsection, we analyze several forms of the reverse function and derive the optimal workload by minimizing the drift-minus-utility. Now we discuss the properties of the derived workload, or the dynamic pricing scheme.

1) Contraction of Queue Dynamics: Based on the utility $\overline{u}[n]$ in Eq. (34) and the drift-minus-utility $f(a[n])$ in Eq. (38), the optimal workload $a^*[n]$ is determined by the value of queue $\overline{q}[n]$. If $\overline{q}[n]$ converges to a steady value, then $a^*[n]$ will converge as well. In this case, the optimal price $p^*[n]$ set by the operator will not change over time. From Eq. (32), the queue dynamics follows $\overline{q}[n+1] = m_1 a[n] + 2m_2\overline{q}[n]$ where m_1, m_2 are in Eq. (31). We next show that this dynamics satisfies the contraction mapping condition under the derived optimal workload, namely $\overline{q}[n]$ will converge to a steady value. Please refer to Appendix E-A in supplementary material for the detailed proof of the following proposition.

Proposition 2: The dynamics of $\overline{q}[n]$ satisfies the contraction mapping condition if Eq. (41) *holds and* $\frac{m_1}{T-m_1} < 1$.

Note that the convexity condition Eq. (41) holds for different forms of $h(a[n])$, like linear, quadratic and logarithm functions. Besides, $\frac{m_1}{T-m_1} < 1$ is also a mild condition, because $m_1 = \frac{e^{\sqrt{\eta b}T} - e^{-\sqrt{\eta b}T}}{\sqrt{\eta b}e^{\sqrt{\eta b}T} + \sqrt{\eta b}e^{-\sqrt{\eta b}T}}$, and usually $T \gg m_1$. Since the optimal price $p^*[n]$ would approach a fixed value, our dynamic pricing scheme essentially implies that the service price changes gradually and will become stable eventually.

2) Utility Vs. Importance Weight: Under the contraction mapping, we have $\overline{q}[n+1] = 2m_2\overline{q}[n] + m_1 a^*[n] = \overline{q}[n]$, thus $\overline{q}[n] = \frac{m_1}{1-2m_2}a^*[n]$ for large value of n. As $\overline{q}[n]$ converges, the utility $\overline{u}[n]$ would also reach a steady value. In [34], its utility will increase if the importance weight V increases, since more emphasis is on the utility term and the utility is decided only by the actions in the Lyapunov framework. However, our utility $\overline{u}[n]$ depends on both the action $a[n]$ and the state $\overline{q}[n]$, which will lead to a different utility behavior in terms of V. We rewrite the queue $\overline{q}[n]$ using the optimal workload $a^*[n]$ and regard $a^*[n]$ as a function of V based on Eq. (40), to illustrate the utility trend over the importance weight below.

Proposition 3: Suppose the dynamics of $\overline{q}[n]$ satisfies the contraction mapping condition and Eq. (41) *holds. Let* $V^* = \frac{1}{\frac{a^*[n]}{b} + \frac{2\eta m_1^2 a^*[n]}{(1-2m_2)^2}}$. *The steady value of $\overline{u}[n]$ will increase in* $V \in (0, V^*)$ *and will decrease in* $V \in (V^*, \infty)$.

See Appendix E-B in supplementary material for the proof. From Proposition 3, if we have $V = V^*$, then the maximum service utility is obtained accordingly.

Remark: The distinct behaviors of queue dynamics and utility trend from classical Lyapunov framework is raised by the fact that the utility maximization is a dynamic programming problem. Hence, our analysis can also provide a new angle for handling such problems in alternative future scenarios.

E. Workload Sojourn Time

As stated in Proposition 2, the queue $\overline{q}[n]$ will reach a steady value under a broad and mild condition. Subsequently, we are dedicated to provide the average sojourn time of the queued workload in the edge computing system, i.e., the delay of users' offloading request. Based on Little's law, the average sojourn time $\overline{d}$ satisfies:

$$a^*[n]\overline{d} = \frac{1}{T}\int_0^T \overline{q}(t)dt. \tag{42}$$

Using the closed-form expression in Eq. (23) and relation $\overline{q}[n] = \frac{m_1}{1-2m_2}a^*[n]$ when queue is steady, we have:

$$\overline{d} = \frac{1}{T}\Big(\frac{1}{\sqrt{\eta b}} + \frac{m_1^2}{1-2m_2} - \frac{2m_2}{\eta b}\Big), \tag{43}$$

where m_1, m_2 are defined in Eq. (31). Actually, the average sojourn time $\overline{d}$ declines over the congestion parameter η, which is also verified in Fig. 10 in later performance evaluations.

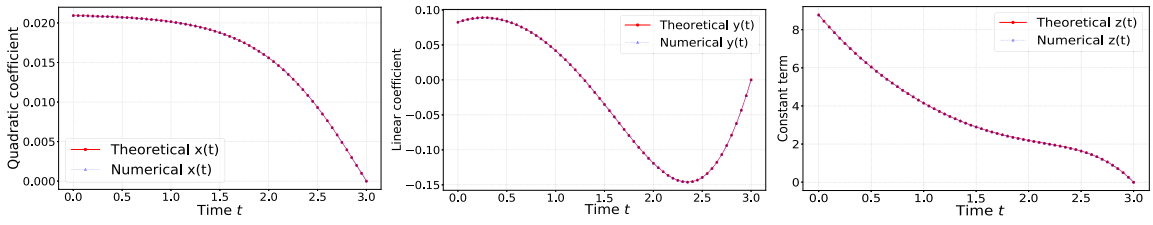
Knowledge of the sojourn time $\overline{d}$ helps end users in making their offloading decisions to edge servers. Specifically, when running delay-sensitive applications, users will not offload workloads if the transmission time and sojourn time are larger than the local processing time, and vice versa. Hence, the operator can commit the service level agreement for providing edge computing services with guaranteed delay. Since this paper mainly studies the workload scheduling from the perspective of edge servers, how to determine user-centered offloading decisions will be our future work.

V. Performance Evaluation

In this section, we evaluate the performance of the workload scheduling and service pricing, especially the mean field approximation and Lyapunov optimization.

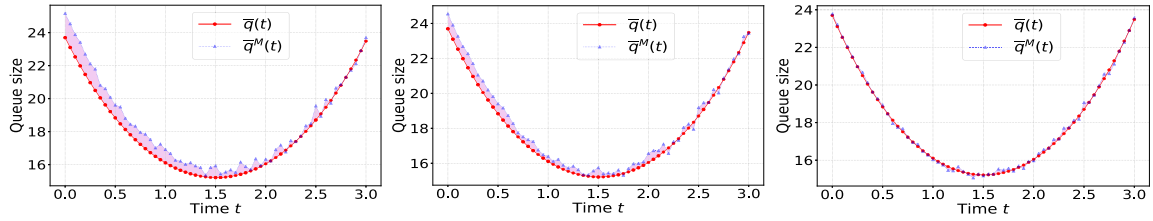
A. Parameter Setting

As described in Section II, each real-world offloading request can be identified by the task size and required service time (CPU cycles), which are also denoted as workloads.



(a) $x(t)$ vs. t (b) $y(t)$ vs. t (c) $z(t)$ vs. t

Fig. 2. Cost coefficients over time t.



(a) Queue size: $M = 50$ (b) Queue size: $M = 100$ (c) Queue size: $M = 200$

Fig. 3. Approximation error of queue size.

When users offload computation to edge servers, a unit computation workload requires about 2000 CPU cycles with the computing capacity of a server measured by GHz [16]. Besides, the energy consumption coefficient is 10^{-26} according to real measurements [20]. Therefore, for the scheduling cost in Eq. (3), parameter α should be $10^{-26} \times (10^9 \times 2000)^2 = 0.04$. Moreover, β and γ are set to 0.02 and 0.03, respectively, since unit transmission cost and unit processing cost at the cloud datacenter are usually smaller. Note that $\alpha < \beta + \gamma$, which is consistent with the fact that edge servers prefer processing workloads locally over offloading remotely. The values of η, λ are set to $0.01, 0.01$ accordingly.

The initial queue size $q_i[1]$ is randomly sampled in $[10, 30]$, thus the expected value is $\overline{q}[1] = \mathbb{E}[q_i[1]] = 20$. In line with the queue size, we consider that the randomness magnitude σ_i^2 is uniformly valued in $[0, 4]$ with $\overline{\sigma}^2 = 2$ to model an unbalanced arrival rate $a_i(t)$ among edge servers. Finally, let the slot duration T be 3.

B. Mean Field Approximation

1) Validation of Cost Coefficients: We first validate the accuracy of our derived quadratic cost coefficients $x(t), y(t), z(t)$ in Lemmas 1-3. We leverage the tool, *scipy.integrate.odeint* in Python, to numerically solve ODEs of Eqs. (16)-(19). Specifically, we discretize one time slot, i.e., duration $T = 3$, into 60 small timesteps with one stepsize of 0.05, and plot both theoretical and numerical results in Fig. 2. One can see that our theoretical derivations are perfectly matched with the numerical results. Therefore, the correctness of obtained expressions for $x(t), y(t), z(t)$ is verified.

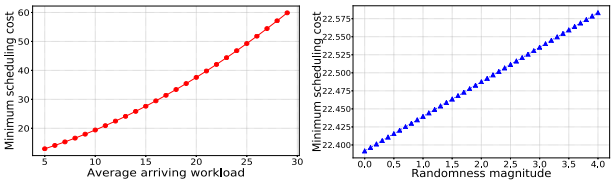
2) Approximation Error of Queue Size: We compare the average queued workload $\overline{q}^M(t)$ in Eq. (2) and its approximation $\overline{q}(t)$ in Eq. (23) when the number of edge servers M changes. Considering the stochastic arrival rate $a_i(t)$, we first numerically compute each queue $q_i(t)$ subject to the queue dynamics in Eq. (1), and then obtain the corresponding $\overline{q}^M(t)$. Varying the value of M, we show the obtained $\overline{q}(t)$ in the mean field model and the average queued workload $\overline{q}^M(t)$ in Fig. 3. When $M = 50$, the error between $\overline{q}(t)$ and $\overline{q}^M(t)$ is relatively large. As M increases to 100, the error becomes smaller. For $M = 200$, the error vanishes, and $\overline{q}^M(t)$ converges to $\overline{q}(t)$. Therefore, the approximation error decreases as M increases.

3) Minimum Scheduling Cost: Let us discuss the minimum scheduling cost $c[n]$ expressed in Eq. (28). We show how $c[n]$ changes with respect to the average arriving workload $a[n]$ and the randomness magnitude σ^2 in Fig. 4. The results demonstrate that the minimum cost $c[n]$ is expected to increase over $a[n]$ and σ^2. Combining with Lemmas 2 and 3, we know that $c[n]$ is a quadratic and linear function pertaining to $a[n]$ and σ^2, respectively, which are consistent with results in Fig. 4.

4) Heterogeneous Arriving Workload: Now we exhibit the workload scheduling under heterogeneous workloads. From Section III-B, the quadratic coefficient $x_i(t)$ and queue $\overline{q}(t)$ are the same as those under the uniform arriving workload, while the terms $y_i(t)$ and $z_i(t)$ are different. Fig. 5(a) shows $y_i(t)$ and $z_i(t)$ corresponding to a server whose arriving rate $a_i[n]$ is different from the average value $\overline{a}[n]$. Though the exact values are distinct, the cost coefficients have similar shapes to the uniform case in Figs. 2(b)-2(c). This observation also holds pertaining to the scheduling cost $c_i[n]$ over $a_i[n]$ in Fig. 5(b), which is akin to the trend in Fig. 4(a). Thereofre, the uniform workload model can give us insights for designing decentralized workload scheduling in a more complex scenario, where we can enjoy the merits of more subtle analysis using closed-form cost expressions. Considering this, later evaluations are largely based on the uniform workload model.

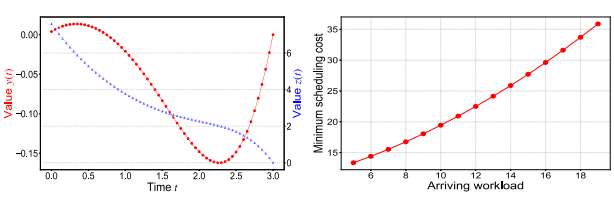
C. Lyapunov Optimization

W.l.o.g., we leverage two forms of the reverse function $h(a[n])$: 1) linear function $h(a[n]) = -2a[n] + 50$ which is



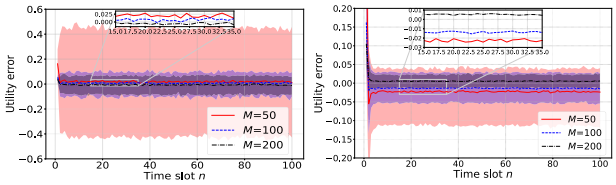
(a) Average arriving workload (b) Randomness magnitude

Fig. 4. Trend of minimum scheduling cost.



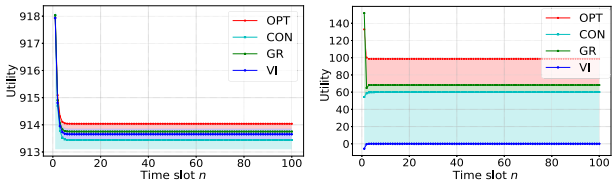
(a) Cost coefficients (b) Average arriving workload

Fig. 5. Heterogeneous cost coefficients.



(a) Linear function (b) Logarithm function

Fig. 6. Approximation error of deterministic service utility.



(a) Linear function (b) Logarithm function

Fig. 7. Utility comparison results.

concise but representative; 2) logarithm function $h(a[n]) = \ln \frac{300}{1+a[n]}$ since many demand-price curves are fitted in exponential function, like for smart grid [26] and ride sharing [27] which can be a potential application of edge computing. Actually, other reverse functions are also applicable here, as mentioned in Section II-C. We will obtain the optimal workload (price) using these two functions in line with Section IV-C, and compute the service utility accordingly.

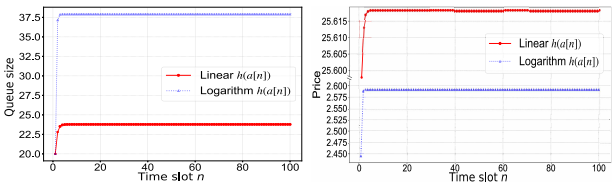
1) Approximation Error of Deterministic Service Utility: The equivalent problem in Theorem 3 is based on the condition that $M \to \infty$. As for finite value of M, we first use the Lyapunov optimization framework to calculate the optimal workload $a^*[n]$ for the linear and logarithm $h(a)$, respectively, and then substitute the obtained $a^*[n]$ into the average utility calculation for the M-edge server system. We run the simulation for six times, and plot the average approximation error of the deterministic service utility in Fig. 6. One can observe that the error is very small, and very close to 0 as M increases. For instance, the errors when $M = 200$ for both two forms of $h(a[n])$ are nearly 0. Therefore, the deterministic service utility in Eq. (34) is equivalent to the original average utility in Eq. (8) for large value of M. In addition, one can also observe that the error variances, which are envelopes padded with light colors, shrink as M increases. Hence, the variance fluctuation of the average utility weakens when M is large.

2) Utility Comparison: Next, we compare the results for our proposed Lyapunov optimization with other approaches using the equivalent deterministic service utility. We denote the Lyapunov optimization as OPT, and choose the following three baseline methods for comparisons to show that minimizing the drift-minus-utility can achieve a larger long-term utility.

- CON: constant workload. The operator adopts a fixed price over time slots, so that the average arriving workload is also constant. Based on the implemented $h(a[n])$ functions, the constant workloads are set to 12.5 and $\ln 300$ for linear and logarithm $h(a[n])$, respectively.
- GR: greedy algorithm. GR greedily maximizes the one-slot utility without considering the need to push the queue size to a low level.
- VI: value iteration [32]. The utility maximization of Eq. (33) is a dynamic programming problem. VI iteratively updates the utility $\overline{u}[n]$ and the action $a[n]$ in each time slot n.

We consider the whole process runs for 100 time slots in total, and obtain the utility $\overline{u}[n]$ of the above four methods in each time slot. The results for linear and logarithm $h(a[n])$ are illustrated in Fig. 7. For both two forms of $h(a[n])$, OPT can achieve the highest utility, which is particularly conspicuous for logarithm $h(a[n])$ in Fig. 7(b). The reason is that OPT will simultaneously minimize the queue size and maximize the immediate utility. As for CON, the utility is small since it is not adaptive. GR mainly focuses on one-slot maximization of the utility, so it behaves myopically. In fact, GR often has a higher utility in the initial few time slots, and then its utility will decrease and perform worse than OPT. Regarding VI, since the utility $\overline{u}[n]$ is non-positive definite, it may not converge to the optimal result. Therefore, the performance of VI is worse than OPT, which is obvious for logarithm $h(a[n])$ in Fig. 7(b). To sum up, OPT can guarantee a larger long-term utility by minimizing the drift-minus-utility.

3) Convergence of Queue and Price Dynamics: Under the contraction mapping, the dynamics of the queue $\overline{q}[n]$ will converge to a steady value. As discussed in Section IV-C, linear and logarithm $h(a)$ satisfy the convexity condition in Eq. (41). Moreover, since the slot duration $T = 3$, and then m_1 in Eq. (31) is 1.438, so that $\frac{m_1}{T-m_1} = 0.921 < 1$. According to Proposition 2, the dynamics of $\overline{q}[n]$ satisfies the contraction mapping condition. To show this, we plot the queue dynamics over 100 time slots for two forms of $h(a[n])$ in Fig. 8(a). One can see that $\overline{q}[n]$ quickly converges to a steady value after a few time slots, and this implies that the optimal price will also remain steady, which is displayed in Fig. 8(b).



(a) Queue dynamics (b) Service price

Fig. 8. Queue and price.

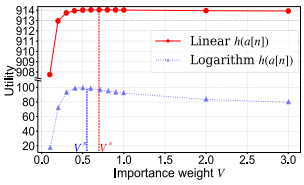


Fig. 9. Steady $\overline{u}[n]$ vs. V.

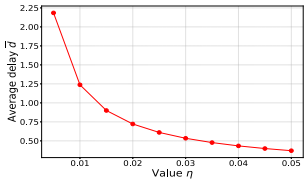
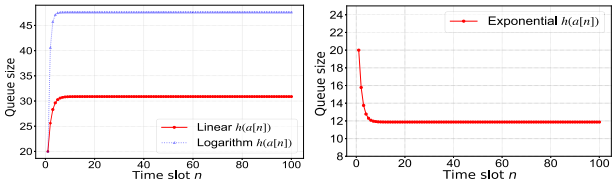


Fig. 10. Sojourn time.



(a) Queue, $T = 2$ (b) Queue, exponential function

Fig. 11. Demonstration of robust convergence.

4) Utility vs. V: Now we show the change of the service utility $\overline{u}[n]$ for different values of importance weight V. If the queue dynamics satisfies the contraction mapping condition, the utility $\overline{u}[n]$ will approach a steady value as well. Proposition 3 states that the steady value of $\overline{u}[n]$ first goes up and then goes down as V increases. We display this trend in Fig. 9 for two forms of $h(a[n])$, and we observe that $\overline{u}[n]$ will first increase and then decrease. Therefore, one can set V to the value of V^* in Proposition 3 to get the maximum utility. From the figure, V^* is around 0.7 for linear $h(a[n])$ and around 0.55 for logarithm $h(a[n])$.

5) Sojourn Time: The sojourn time $\overline{d}$ reflects the average latency of workload scheduling, which is tightly dependent on the queue length or the congestion parameter η in Eq. (3). Eq. (43) indicates that the delay will decrease if we put more weight on the congestion term, as justified in Fig. 10 since $\overline{d}$ tends to decline over η.

6) Robust Convergence: Previously, we demonstrate that the queue $\overline{q}[n]$ converges to a steady value for linear and logarithm functions when $T = 3$, namely the convexity condition and contraction mapping are satisfied. In fact, they are only sufficient conditions, and even violated, the queue may still converge as well. Let $T = 2$, and then m_1 in Eq. (31) is 1.3, so that $\frac{m_1}{T-m1} = 1.857 > 1$. The queue dynamics for both two forms of $h(a[n])$ is drawn in Fig. 11(a), which illustrates that the queue still reaches a steady value even the contraction mapping does not hold. To further verify that Lyapunov optimization is robust in terms of queue convergence, we exhibit the queue dynamics for the exponential function $h(a[n]) = 20e^{-0.2a[n]}$, which might not be convex as $\overline{x} = 0.0311 < \frac{1}{2}TC_{\max}C_pe^{-3} = 0.1991$ when $T = 2$. Fig. 11(b) shows the change of $\overline{q}[n]$ for the exponential function, and it indicates that the queue still converges even both the conditions of contraction mapping and convexity are not satisfied. In general, convergence of queue $\overline{q}[n]$ to a steady value is observed in different scenarios, i.e., Lyapunov optimization has robust convergence.

7) Robust Utility: At last, we show that our proposed Lyapunov optimization based pricing scheme, OPT, still has satisfactory performances even the reverse function $h(a[n])$ is not a contraction or convex mapping. Specifically, Fig. 12 displays the utility of OPT and baseline methods (CON, GR, VI) when $T = 2$ for linear, logarithm and exponential $h(a[n])$, respectively. The results justify that OPT can attain a higher service utility compared to these benchmarks.

VI. Related Work

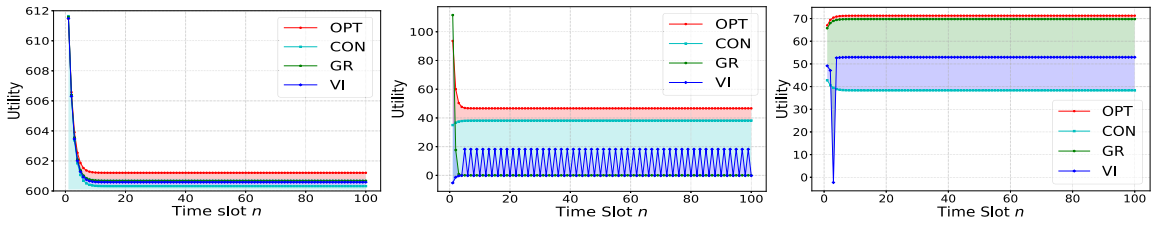
In this paper, we study the optimal workload scheduling and dynamic service pricing in a large-scale edge computing system through the mean field game. In the following, we briefly survey the related works.

A. Edge Computing Offloading

In edge computing, great efforts have been devoted to workload offloading [5], [6]. Chen *et al.* propose a response updating method to calculate the equilibrium offloading decision, where actions are taken by end users instead of edge servers [15]. On this basis, Zheng *et al.* exploit the dynamic computation offloading from the perspective of end users, and each stage is modeled as a potential game [36]. Li *et al.* design a cooperative task placement model in edge computing with and without task deadlines, where the long-term social cost is minimized via Lyapunov framework [37]. Besides, a two time-scale Lyapunov optimization based workload scheduling and server management scheme is studied [38], yet both timescales are pertaining to discrete time slots. If considering the cooperation between edge servers, distributed edge sampling can also be incorporated [40]. Load balancing is essential to avoid overloaded queues [39]. Beraldi *et al.* study load balancing in edge computing to reduce the service blocking probability and the service delay [19]. Especially for the case where edge servers are geo-distributed, load balancing can maintain comparable workloads across these servers, and ensure the fair usage of cloud resource [8]. However, previous works can not handle the workload offloading efficiently when there are many servers in a large-scale edge computing system [13].

B. Mean Field Game

Mean field game is a statistical method to deal with interactions among a large population of agents, which is



(a) Linear function $T = 2$ (b) Logarithm function $T = 2$ (c) Exponential function $T = 2$

Fig. 12. Demonstration of robust utility.

proposed by Lasry *et al.* [41] and Huang *et al.* [42]. As for edge computing, Banez *et al.* apply the mean field game to analyze the computation offloading among multi-access points, where the running cost is dependent on the expected network state and offloading control [29]. Furthermore, Kim *et al.* use the mean field game to determine the caching strategy in ultra-dense networks for reducing the long-term cost [44]. Nevertheless, most existing works need to rely on numerical methods, rather than deriving explicit results, to evaluate the performance of their formulated mean field models.

C. Service Pricing

Pricing scheme is of paramount importance in terms of providing service to end users, such as the data plan in wireless networks [45]. For edge computing, Zhao *et al.* model the computation offloading between users and access points as a stochastic game, where access points will set prices to maximize their profits and users would devise the offloading strategies to reduce latency and charged fee [12]. Besides, Liu *et al.* formulate a Stackelberg game to characterize users' computation offloading to an edge server, and further determine the optimal prices for the server to maximize its revenue [46]. Li *et al.* propose an online truthful double auction-based open edge market that facilitates individual servers to cooperatively provide services while economizing their valuable resources [47]. Conventional pricing schemes for edge computing are usually too complicated to be implemented, as a uniform service price is more practical in real-world applications.

We investigate the workload scheduling and service pricing for geo-distributed edge computing systems, where load balancing among many edge servers is considered. Specifically, we derive a closed-form cost in a decentralized manner based on the mean field game, and obtain a large long-term utility through designing a dynamic pricing scheme.

VII. Conclusion

In this paper, we study the optimal scheduling and dynamic pricing for a large-scale edge computing system. We propose a two-timescale optimization framework to handle actions occurring at different timescales. By applying the mean field game, we derive the local optimal processing and offloading speeds at the small timescale for each edge server, and obtain a closed-form scheduling cost which has $O\left(\frac{1}{\sqrt{M}}\right)$ error with M, the number of edge servers. Using the explicitly derived cost, we design a dynamic pricing scheme at the large timescale based on the Lyapunov optimization framework. Therefore, we can improve the long-term service utility by minimizing the drift-minus-utility in each time slot. Finally, we carry out extensive evaluations to demonstrate the accuracy of our mean field model, and the superiority of the designed pricing scheme when compared with the state-of-the-art methods.

References

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

[2] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service entity placement for social virtual reality applications in edge computing," in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 468–476.

[3] Q. Yuan, H. Zhou, J. Li, Z. Liu, F. Yang, and X. Shen, "Toward efficient content delivery for automated driving services: An edge computing solution," *IEEE Netw.*, vol. 32, no. 1, pp. 80–86, Jan./Feb. 2018.

[4] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[5] Y. Jararweh, A. Doulat, O. Al-Qudah, E. Ahmed, M. Al-Ayyoub, and E. Benkhelifa, "The future of mobile cloud computing: Integrating cloudlets and mobile edge computing," in *Proc. IEEE ICT*, May 2016, pp. 1–5.

[6] 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.

[7] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, and B. Ottenwälder, "Incremental deployment and migration of geo-distributed situation awareness applications in the fog," in *Proc. ACM DEBS*, 2016, pp. 258–269.

[8] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun. 2019.

[9] Y. Liu, C. Xu, Y. Zhan, Z. Liu, J. Guan, and H. Zhang, "Incentive mechanism for computation offloading using edge computing: A Stackelberg game approach," *Comput. Netw.*, vol. 129, pp. 399–409, Dec. 2017.

[10] D. Zhang *et al.*, "Near-optimal and truthful online auction for computation offloading in green edge-computing systems," *IEEE Trans. Mobile Comput.*, vol. 19, no. 4, pp. 880–893, Apr. 2020.

[11] Q. Wang, S. Guo, J. Liu, C. Pan, and L. Yang, "Profit maximization incentive mechanism for resource providers in mobile edge computing," *IEEE Trans. Serv. Comput.*, vol. 15, no. 1, pp. 138–149, Feb. 2022.

[12] Z. Zhao, W. Zhou, D. Deng, J. Xia, and L. Fan, "Intelligent mobile edge computing with pricing in Internet of Things," *IEEE Access*, vol. 8, pp. 37727–37735, 2020.

[13] L. Tawalbeh, Y. Jararweh, and F. Dosari, "Large scale cloudlets deployment for efficient mobile cloud computing," *J. Netw.*, vol. 10, no. 1, pp. 70–76, 2015.

[14] S. Jošilo and G. Dán, "Decentralized algorithm for randomized task allocation in fog computing systems," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 85–97, Feb. 2019.

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

[16] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3887–3901, Dec. 2016.

[17] Z. Zheng and N. B. Shroff, "Online multi-resource allocation for deadline sensitive jobs with partial values in the cloud," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.
[18] H. Gao, W. Li, R. A. Banez, Z. Han, and H. V. Poor, "Mean field evolutionary dynamics in dense-user multi-access edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 7825–7835, Dec. 2020.
[19] R. Beraldi, A. Mtibaa, and H. Alnuweiri, "Cooperative load balancing scheme for edge computing resources," in *Proc. IEEE FMEC*, May 2017, pp. 94–100.
[20] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.
[21] V. Gupta, A. F. Dana, J. P. Hespanha, R. M. Murray, and B. Hassibi, "Data transmission over networks for estimation and control," *IEEE Trans. Autom. Control*, vol. 54, no. 8, pp. 1807–1819, Aug. 2009.
[22] M. Larrañaga, U. Ayesta, and I. M. Verloop, "Index policies for a multi-class queue with convex holding cost and abandonments," in *Proc. ACM SIGMETRICS*, 2014, pp. 125–137.
[23] L. Zhang, W. Wu, and D. Wang, "Time dependent pricing in wireless data networks: Flat-rate vs. usage-based schemes," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 700–708.
[24] Z. Xiong, J. Kang, D. Niyato, P. Wang, and H. V. Poor, "Cloud/edge computing service management in blockchain networks: Multi-leader multi-follower game-based ADMM for pricing," *IEEE Trans. Serv. Comput.*, vol. 13, no. 2, pp. 356–367, Apr. 2020.
[25] D. T. Nguyen, L. B. Le, and V. Bhargava, "Price-based resource allocation for edge computing: A market equilibrium approach," *IEEE Trans. Cloud Comput.*, vol. 9, no. 1, pp. 302–317, Jan./Mar. 2021.
[26] R. Yu, W. Yang, and S. Rahardja, "A statistical demand-price model with its application in optimal real-time price," *IEEE Trans. Smart Grid*, vol. 3, no. 4, pp. 1734–1742, Dec. 2012.
[27] Z. Fang, L. Huang, and A. Wierman, "Prices and subsidies in the sharing economy," in *Proc. ACM WWW*, 2017, pp. 53–62.
[28] W. Zhang, Z. Zhang, S. Zeadally, H.-C. Chao, and V. C. M. Leung, "Energy-efficient workload allocation and computation resource configuration in distributed cloud/edge computing systems with stochastic workloads," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1118–1132, Jun. 2020.
[29] R. A. Banez *et al.*, "Mean-field-type game-based computation offloading in multi-access edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 8366–8381, Dec. 2020.
[30] E. Todorov, "Optimal control theory," in *Bayesian Brain: Probabilistic Approaches to Neural Coding*. Cambridge, MA, USA: MIT Press, 2006, pp. 269–298.
[31] Z. Wang, J. Ye, and J. C. S. Lui, "An online mean field approach for hybrid edge server provision," in *Proc. ACM MobiHoc*, 2021, pp. 131–140.
[32] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 943–949, Jun. 2008.
[33] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 3, pp. 621–634, Mar. 2014.
[34] X. Wang, R. Jia, X. Tian, and X. Gan, "Dynamic task assignment in crowdsensing with location awareness and location diversity," in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 2420–2428.
[35] M. J. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan & Claypool, 2010.
[36] J. Zheng, Y. Cai, Y. Wu, and X. Shen, "Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 771–786, Apr. 2018.
[37] Y. Li *et al.*, "Cooperative service placement and scheduling in edge clouds: A deadline-driven approach," *IEEE Trans. Mobile Comput.*, vol. 21, no. 10, pp. 3519–3535, Oct. 2022.
[38] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. Neely, "Data centers power reduction: A two time scale approach for delay tolerant workloads," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1421–1429.
[39] M. Alizadeh *et al.*, "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proc. ACM Conf. SIGCOMM*, Aug. 2014, pp. 503–514.
[40] L. Jia, Q.-S. Hua, H. Fan, Q. Wang, and H. Jin, "Efficient distributed algorithms for holistic aggregation functions on random regular graphs," *Sci. China Inf. Sci.*, vol. 65, no. 5, pp. 1–19, May 2022.
[41] J.-M. Lasry and P.-L. Lions, "Mean field games," *Jpn. J. Math.*, vol. 2, no. 1, pp. 229–260, 2007.
[42] M. Huang, R. P. Malhamé, and P. E. Caines, "Large population stochastic dynamic games: Closed-loop McKean–Vlasov systems and the Nash certainty equivalence principle," *Commun. Inf. Syst.*, vol. 6, no. 3, pp. 221–252, 2006.
[43] S. Benachour, B. Roynette, D. Talay, and P. Vallois, "Nonlinear self-stabilizing processes—I existence, invariant probability, propagation of chaos," *Stochastic Processes Appl.*, vol. 75, no. 2, pp. 173–201, Jul. 1998.
[44] H. Kim, J. Park, M. Bennis, S.-L. Kim, and M. Debbah, "Mean-field game theoretic edge caching in ultra-dense networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 935–947, Jan. 2020.
[45] L. Zhang, W. Wu, and D. Wang, "Sponsored data plan: A two-class service model in wireless data networks," in *Proc. ACM SIGMETRICS*, Jun. 2015, pp. 85–96.
[46] M. Liu and Y. Liu, "Price-based distributed offloading for mobile-edge computing with computation capacity constraints," *IEEE Wireless Commun. Lett.*, vol. 7, no. 3, pp. 420–423, Jun. 2018.
[47] Y. Li, H. C. Ng, L. Zhang, and B. Li, "Online cooperative resource allocation at the edge: A privacy-preserving approach," in *Proc. IEEE ICNP*, Oct. 2020, pp. 1–11.

Xiong Wang (Member, IEEE) received the B.E. degree in electronic information engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2014, and the Ph.D. degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2019. He was a Post-Doctoral Fellow at the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China, from 2019 to 2021. He is currently an Associate Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. His research interests include distributed machine learning systems, federated learning, network flow control, mean field analysis, and cloud/edge computing.

Jiancheng Ye (Member, IEEE) received the B.E. degree in network engineering from Sun Yat-sen University, Guangzhou, China, in 2008, the M.Phil. degree in computer science and engineering from The Hong Kong University of Science and Technology, Hong Kong, in 2011, and the Ph.D. degree in computer networking from The University of Hong Kong, Hong Kong, in 2018. He was a Software Engineer with Harmonic Inc., from 2011 to 2014, and a Post-Doctoral Fellow at The University of Hong Kong, from 2018 to 2019. He is currently a Researcher with the Network Technology Laboratory, Huawei, Hong Kong. His research interests include congestion control, queue management, optimization of computer networks, edge computing, and online learning. He is a member of ACM.

John C.S. Lui (Fellow, IEEE) received the Ph.D. degree in computer science from the University of California at Los Angeles. He is currently the Choh-Ming Li Chair Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His current research interests include machine learning, online learning (e.g., multi-armed bandit, reinforcement learning), network Science, future internet architectures and protocols, network economics, network/system security, large scale storage systems. He is an Elected Member of the IFIP WG 7.3, a Fellow of ACM, a Senior Research Fellow of the Croucher Foundation, and was the Chair of the ACM SIGMETRICS from 2011 to 2015. He received various departmental teaching awards and the CUHK Vice-Chancellor's Exemplary Teaching Award. He was a co-recipient of the Best Paper Award in the IFIP WG 7.3 Performance 2005, IEEE/IFIP NOMS 2006, SIMPLEX 2013, and ACM RecSys 2017.