Online Queue-Aware Service Migration and Resource Allocation in Mobile Edge Computing

An Du, Jie Jia, *Member, IEEE*, Jian Chen, *Member, IEEE*, Xingwei Wang, and Min Huang, *Member, IEEE*

***Abstract*—Mobile edge computing (MEC) integrated with Network Functions Virtualization (NFV) helps run a wide range of services implemented by Virtual Network Functions (VNFs) deployed at MEC networks. This emerging paradigm offers flexible edge resource management for Internet Service Providers (ISPs) and improves service satisfaction of diverse applications. However, a critical challenge lies in processing extremely unpredictable and bursty traffic flow, especially in event-trigger sensing tasks. In this paper, we study queue-aware service migration and resource allocation in an MEC network, where different devices with different arrival patterns request different network services. To this end, we first formulate a novel long-term operation cost minimization problem subject to the stability of multiple queues by adopting a simultaneous migration scheme and determining resource allocation dynamically. To address the problem efficiently, we propose an online control framework to make decisions without any prior system information. We first adopt the Lyapunov optimization technique to decompose the long-term optimization problem into multi-per-slot subproblems. Then, we adopt a decoupled optimization algorithm based on the coordinate descent method and augmented Lagrangian method to solve the offline mixed integer non-linear programming (MINLP) subproblems. Extensive results reveal that the proposed online optimization framework can efficiently balance queue stability and cost consumption.**

***Index Terms*—Bursty traffic, Lyapunov optimization, mobile edge computing, online control, virtual network functions.**

I. Introduction

NOWADAYS, the rapid growth of the Internet of Things (IoT) and wireless communication technologies provide industries with new opportunities to develop various emerging applications [1]. The generated volumes of IoT data pose a great challenge for IoT devices with limited computing and battery capabilities [2]. Traditionally, IoT platforms make use of cloud services in remote datacenters located far away from IoT devices for data storage and processing. However, cloud computing meets serious bottlenecks due to unacceptable communication delays posed by long-distance transmission and the unprecedented risk of congestion caused by the transmission of massive amounts on backbone networks. These issues ultimately lead to a degradation in the quality of IoT applications. As a remedy to these challenges, Mobile Edge Computing (MEC) has been regarded as a promising paradigm to provide higher service quality and energy-conserving task execution for IoT devices by shifting the computing ability from the centralized cloud to the network edge [3].

Nevertheless, the extensive service diversity and the limited resources pose a new challenge for MEC networks to deploy and manage services flexibly and energy-efficiently. To satisfy the growing demands of Internet Service Providers (ISPs) and IoT applications, Network Functions Virtualization (NFV) has been one of the most critical technologies to embrace excellent flexibility, agility, and efficiency by decoupling network functions from hardware [4]. Significant cost savings can also be achieved since a wide range of services such as mobile core, deep packet inspection (DPI), security and routing can be implemented handily through the provisioning of a collection of Virtual Network Functions (VNFs) [5]. Driven by the mature implementation of NFV, the ISP in MEC can handle the traffic efficiently by mapping the ever-increasing requirements for various IoT applications into the requested resources on demand [6], [7]. As a result, the Quality of Service (QoS) of IoT devices can be significantly enhanced.

To process heterogeneous computing tasks, a collection of corresponding VNFs should be placed on demand. Unlike cloud datacenters, virtually-enabled edge servers are characterized by low computation and storage capacity. This poses significant challenges in scalable computing capabilities and customed QoS performance guarantee [8]. In addition, the ever-changing service requirements and dynamic communication environment make the network suffer from spatially uneven traffic distribution. To enable a load-balanced guarantee, edge-edge cooperation has become a promising way to provide distributed resources [9]. Therefore, the potential of edge-edge cooperation needs to be fully released by jointly optimizing service placement and resource allocation in the NFV-enabled MEC network.

Received 14 March 2024; revised 26 November 2024 and 27 December 2024; accepted 27 December 2024. Date of publication 7 January 2025; date of current version 20 May 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 92167103, Grant 62172084, and Grant 62132004, in part by the NSFC Key Supported Project of the Major Research Plan under Grant 92267206, in part by the Aeronautical Science Foundation of China under Grant 20230026050001, in part by the Young and Middle-Aged Leading Talents in Technological Innovation of Shenyang under Grant RC231173, and in part by the Fundamental Research Funds for the Central Universities under Grant N2416007, Grant N2416001, Grant N2424010-18, Grant N2416006, and Grant N2216009. The review of this article was coordinated by Dr. Tao Dusit Niyato. *(Corresponding author: Jie Jia.)*

An Du, Jie Jia, Jian Chen, and Xingwei Wang are with the School of Computer Science and Engineering, the Engineering Research Center of Security Technology of Complex Network System, Northeastern University, Shenyang 110819, China, and also with the Key Laboratory of Intelligent Computing in Medical Image, Ministry of Education, Northeastern University, Shenyang 110819, China (e-mail: 2110663@stu.neu.edu.cn; jiajie@mail.neu.edu.cn; chenjian@mail.neu.edu.cn; wangxw@mail.neu.edu.cn).

Min Huang is with the College of Information Science and Engineering, State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China (e-mail: mhuang@mail.neu.edu.cn).

Digital Object Identifier 10.1109/TVT.2024.3524747

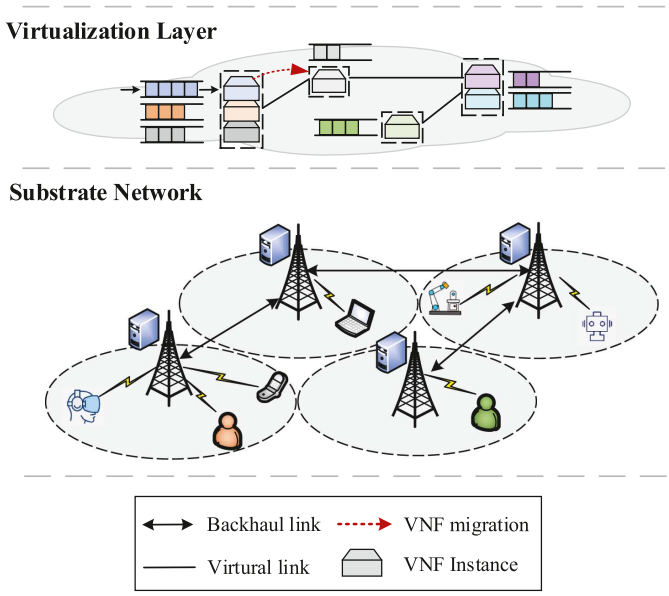

Fig. 1. An illustrative example of an NFV-enabled MEC network.

Most existing works developed service migration/placement strategies for satisfactory service quality by taking the deadline as a hard constraint [8], [9], [10], [11], [12] or devoting to optimizing total delay within each slot [13], [14], [15]. A typical implicit assumption in these works is that the MEC network has an infinite backlog and can process the offloaded data immediately. However, recent advancements in IoT applications bring massive bursty and dynamic traffic flows to the network. Especially in the intelligent healthcare system, wearable devices on patients continuously monitor vital signs, and the traffic load becomes highly unpredictable and significantly increases when a mass casualty event occurs, or someone's condition fluctuates rapidly [16]. The bursty traffic data also comes from intelligent traffic systems, event-trigger sensing tasks, and other online events, posing a critical challenge for the network to process generated data promptly. Consequently, network congestion and the number of backoff times of IoT devices will dramatically increase. Therefore, both queue dynamics and service performance need to be considered to analyze and control the stochastic network with time variation. Furthermore, adaptive and online network management is essential to guarantee stable system operation. Up to now, a few works have investigated the impact of traffic flows with burstiness on the network [17]. For example, queuing dynamics in the network routing control problem [18] and computing offloading in MEC [19], [20], [21] were investigated for promoting network stability control. However, they ignore the fact that tasks can be processed only if the requested services are provisioned first. To this end, joint optimization of dynamic service migration and flexible resource allocation can be seen as a significant solution for supporting diverse service requirements and ensuring network stability, which has not been well studied.

To clarify, Fig. 1 illustrates the queuing dynamics on the VNF instances impacted by the stochastical task data arrival in sequential time slots. Triggered by a significant change in a specific condition, the IoT device produces bursty traffic flow. Assuming that after a while, the corresponding VNF occurs severe queue backlog due to the constraint of resources on the hosted edge server. Benefiting from the edge-edge cooperation, the VNF should be migrated to other edge servers with more available computing resources to stabilize the data queue. Otherwise, the perceived service quality will greatly deteriorate, and the computational performance of other VNFs deployed on the same edge server will be adversely affected accordingly. To reduce the system overhead required for data synchronization, the data queue also should be migrated along with the VNF. However, migration across backhaul links may cause massive network resource occupation and expensive operation costs for ISP. Therefore, deploying VNFs cost-efficiently is also essential for ISP.

In addition to ensuring service quality by dynamic service migration, it is equally important to optimize computing resource allocation on edge servers. This can not only satisfy diverse service requirements for IoT devices but also help the ISP achieve computing energy savings. To this end, we mainly focus on addressing the following challenges: How to dynamically make service migration control decisions and effectively assign computing resources in an MEC system endowed with limited resources? Besides, How to maintain network stability in an energy-efficient manner for the ISP, without knowing data arrivals and wireless channel conditions? To the best of our knowledge, the joint optimization problem has not been investigated for now. The novelty of our work lies in that a highly dynamic NFV-enabled MEC network with irregular and stochastically arrival patterns is investigated. Driven by the distributed resource-sharing capability provided by edge-edge cooperation, joint optimization of service migration and computing resource allocation are adopted as a powerful scheme for saving the energy budget of the ISP and guaranteeing network stability. Aiming to achieve energy cost minimization and maintain network stability over the long run, the Lyapunov optimization theory is adopted since it simultaneously enables stability and performance optimization. To this end, we develop a novel online algorithm based on the Lyapunov optimization theory to transform the time-averaged optimization problem into a queue stability problem. Then, a decoupled framework is proposed to jointly obtain the service migration and computation resource allocation decisions. The main contributions of this paper are summarized as follows.

1) We formulate a novel queue-aware service migration and computing resource allocation problem to minimize the long-term service provision cost of the ISP, subject to the total computing capacity of each edge server and network stability. We also prove that the formulated problem is NP-hard.
2) We propose an online control framework by applying the Lyapunov optimization technique for joint stability and performance optimization. For each per-slot offline problem, we propose a coordinate descent (CD)-based decoupled framework to optimize service migration. Besides, we devise a distributed optimization method based on the augmented Lagrangian method

(ALM) and primal decomposition for optimizing resource allocation.

3) We conduct extensive simulations to evaluate the performance of our proposed framework. Simulation results demonstrate that it can rapidly converge to optimal results and achieve a superior balance between queue stability and cost consumption compared to the other proposed heuristic benchmarks and approximate algorithms.

The remainder of the paper is organized as follows. In Section II, we present a brief overview of the related literature. In Section III, we give the system model and formulate a long-term average energy consumption minimization problem. In Section IV, we develop an online optimization framework and its theoretical analysis. In Section V, we carry out ample simulations and contrastive analysis for our numerical results to verify the effectiveness of the proposed algorithms. Finally, Section VI concludes this paper.

II. Related Work

The great advance in 5G communications and IoT technologies boosts emerging IoT applications with stringent timeliness yet resource-hungry requirements [22]. Although the integration of MEC and NFV alleviates the issues of enhancing resource capacity and facilitating flexibility of resource management, the diversity of applications requires different services to be placed in networks on demand [23]. Meanwhile, resources in networks need to be allocated efficiently to guarantee service quality in a cost-efficient manner. In recent years, service placement and resource allocation have drawn extensive attention in many research works.

A. Service Placement in MEC Networks

The service placement was widely investigated in the service function chain (SFC) embedding problem, i.e., mapping the virtual node/VNF to the physical node. For instance, considering the dynamic service request arrivals, the authors in [24] proposed an online optimization algorithm to balance the SFC embedding cost and network stability. To provide users with satisfactory services and maintain SFC enforcement, several efforts have been attempted on NFV-enabled MEC networks [25]. However, the SFC embedding problem mainly focuses on providing complicated and integrated network services in multi-hop network propagation, while the service placement problem in NFV-enabled MEC networks concentrates more on providing low-latency services implemented by single, consolidated VNF instances [7], thus making the above methodologies cannot be directly applied to our problem.

Recently, a group of existing works paid plenty of attention to service caching/placement in cloud-assisted MEC networks. For example, the authors in [26] studied the joint optimization of service selection, computation resource allocation, and task offloading in a static network. Considering the stochastic task arrivals, the authors in [15] proposed an online service placement and computation resource allocation strategy based on an improved DRL method. Although the works in [15], [26] exceed the MEC's capacity to the remote cloud, relying on a single-edge cloud greatly restricts MEC performance. Benefiting from the interconnectivity among BSs, MEC enables resource-limited edge servers in a federation to develop a wide range of new services. Supported by edge-edge cooperation, some works have been focused on the service placement/migration problem in static environments. For example, the authors in [7] developed efficient heuristics for jointly placing IoT applications and VNFs, while the authors in [27] proposed an efficient decentralized algorithm for service placement in MEC-enabled dense small-cell network.

However, the highly dynamic MEC environment with time-varying user mobility, communication network conditions, and service popularity poses great challenges for the proposed service placement strategies mentioned above. To this end, several works focused on designing an online service placement strategy. Considering the service continuity incurred by user mobility, dynamic service migration has been considered in several previous works. For example, the authors in [10] designed a reinforcement learning (RL)-based microservice coordination algorithm for online decision-making. To overcome the scalability issue posed by a single super agent, the authors in [11] developed an online algorithm based on multi-agent DRL for joint service migration and mobility optimization. By leveraging auto-regression method-based user mobility prediction, the authors in [12] investigated the throughput maximization problem in MEC networks. Moreover, the authors in [28] designed an online spatially-temporally collaborative algorithm for optimizing service placement and task scheduling. Different from existing solutions that assume stable inter-edge-node data transmission, the authors in [29] further investigated the reliability-guaranteed service migration path selection while considering user mobility and inter-edge-node communication stability. However, service placement and resource allocation schemes need to be carefully designed jointly due to the limited computing capacity of edge nodes, yet receive much less attention.

B. Joint Service Placement and Resource Management in Collaborative MEC Networks

To place services and assign MEC resources efficiently, the authors in [30] adopted the VNF replication scheme and proposed a heuristic algorithm in static scenarios. To improve the user experience for adaptive video streaming networks, the authors in [31] dedicated to the collaborative caching and computing scheme in two timescales. Considering the mobility in satellite edge clouds, the authors in [32] proposed a distributed multi-agent-based service migration approach to manage network resources and a priority-based method to solve the resource conflict problem. Benefiting from the reliable communication and efficient task processing, the authors in [33] investigated both the task deployment and the task offloading in the MEC-empowered air-ground integrated network. Considering the constrained coverage and capacity of terrestrial networks, the authors in [34], [35] invoked aerial platforms to constitute a multi-layer edge computing network and investigated the long-term system performance to adapt to the highly dynamic environment. In [35], the authors developed

TABLE I
COMPARISION WITH OTHER SYSTEM MODEL DESIGN

Ref	MEC network	Service placement	Resource allocation	Queue stability	Network costs
[19]–[21]	single edge node	×	✓	data queue stability	✓
[15], [26]	edge-cloud cooperation	✓	✓	×	×
[7], [27]	static edge collaboration	✓	×	×	✓
[10], [11], [28]	dynamic edge collaboration	✓	×	×	✓
[30]	static edge collaboration	✓	✓	×	✓
[32]	satellite edge clouds	✓	✓	×	✓
[33], [35]	air-ground MEC network	✓	✓	×	✓
[36]	edge-cloud cooperation	✓	✓	task queue stability	×
[37]	vehicle edge network	×	✓	task queue stability	×
Ours	dynamic edge collaboration	✓	✓	data queue stability	✓

a novel multi-time-scale RL-based service placement, network selection, and computation offloading optimization algorithm to guarantee latency and energy requirements, while the dynamic service placement was not considered in [34]. However, the works mentioned above were based on the implicit assumption that the offloaded tasks can be processed within the deadline, thus ignoring the impact of data backlog on network stability. Although the authors in [36] studied service placement and computing resource allocation to ensure queue stability of MEC servers, the costs during the service provision process were not well treated. Moreover, similar to the work in [36] that considered unpredictable task distribution, both the task delay requirements and task queue stability were considered in [37]. However, a common assumption in these works was that each task has a stable data load to be processed within each slot, which is impractical for stochastic environment with burst task data.

C. Comparison Analysis

Unlike all these works mentioned above, we consider a dynamic NFV-enabled MEC network with spatially uneven workloads posed by traffic burstiness. By leveraging edge-edge cooperation, we study the joint optimization of service migration and resource allocation to achieve queue stability energy-efficiently, which has not been investigated hitherto. Table I summarizes the difference between our proposal and the representative system model designs in existing MEC networks. However, the typically unpredictable distribution of traffic arrival and channel conditions makes the problem intricate, causing existing works focusing on offline optimization [7], [26], [27], [38] or prediction schemes [12], [13] to be not applicable. To solve such a stochastic optimization problem, three well-developed methods were proposed for dynamic resource allocation, including the large derivation theory, the Lyapunov optimization, and the Markov decision process (MDP) based approaches [39]. The challenge of utilizing large derivation theory in our problem mainly lies in the limited performance for the general delay regime, and the MDP-based method may exhibit the curse of dimensionality when considered system enlarges. Therefore, we develop an online control framework by adopting the Lyapunov optimization technique without requiring any network information. Next, we propose a CD-based decoupled framework that achieves superior performance compared with heuristics.

TABLE II
IMPORTANT NOTATIONS IN OUR MODEL

Symbol	Definition
$\mathcal{S}$	The set of BSs integrating edge servers, $s \in \mathcal{S}$.
$\mathcal{N}$	The set of IoT devices, $n \in \mathcal{N}$.
$\mathcal{E}$	The set of wired links between BSs, $e_{i,j} \in \mathcal{E}$, $i, j \in \mathcal{S}$.
$w_{i,j}$	The bandwidth of outgoing link $e_{i,j}$.
F_s	The maximum computing capacity of BS s.
τ	The time duration of one time slot.
x_n^s	Indicator of the VNF required by n is placed on s or not.
f_n^s	The assigned computing resources for n.
D_n^a	The amount of sensing results at IoT device n.
D_n^o	The maximum offloading data size of IoT device n.
W	The allocated channel bandwidth.
P_n^{max}	The maximum transmission power of IoT device n.
p	The transmission power of BSs.
h_n	Channel gain between n and its connected base station.
σ^2	Noise power.
A_n	Arrival data size at the VNF serves IoT device n.
D_n	Executed date size of VNF serves IoT device n.
Q_n	The queue length on the VNF serves IoT device n.
ϕ_n	Requested computing cycles per bit of IoT device n.
o_n	The size of VNF instance that serves IoT device n.
E_n^m	Consumed energy of migrating VNF serves IoT device n.
E_n^c	Computing energy consumption for IoT device n.

III. SYSTEM MODEL AND ASSUMPTION

In this section, we introduce the system model of service migration and resource allocation joint optimization problem in an NFV-enabled MEC network. Particularly, we present the corresponding system model and problem formulation in Sections III-A and III-B, respectively. Some key parameter notations are summarized in Table II.

A. System Model

1) NFV-Enabled MEC Network Model: For the service migration and resource allocation in an NFV-enabled MEC network, we consider a hierarchical architecture, including the substrate network and the virtualization layer, as shown in Fig. 1. Each base station (BS) of the MEC network is endowed with a high-performance edge server, which enables load balancing and high-quality services provisioning for IoT devices. We consider the NFV-enabled MEC network as $\mathcal{G} = (\mathcal{S}, \mathcal{N}, \mathcal{E})$, where $\mathcal{S} = \{1, 2, \ldots, S\}$ denotes the set of edge servers, $\mathcal{N} = \{1, 2, \ldots, N\}$ denotes the set of IoT devices, and $\mathcal{E} = \{e_{i,j}, i, j \in \mathcal{S}, i \neq j\}$ denotes the set of backhaul links between ESs [11]. Each $s \in \mathcal{S}$ is an edge server as well as its co-located BS. Multiple network functions requested by IoT devices are implemented by various VNFs running on edge

servers. For example, a network function of data preprocessing is deployed to perform stratified sampling of offloaded data. Also, an intrusion detection system is deployed into one of the edge servers to identify possible intrusions. Physically, edge computing is assumed to be deployed in a fully distributed manner. Each edge server $s \in \mathcal{S}$ provides only limited computing resources to implement VNFs, which is upper bounded by the maximum value F_s (in CPU cycles per second). Because network providers generally operate during a long period, we assume the system runs within a finite time horizon, which is discretized into equal length time slots $\mathcal{T} = \{1, 2, \ldots, T\}$. The time duration of each slot is denoted as τ. Specifically, the whole time slot is allocated for each IoT device and VNF for computation offloading and data processing, respectively.

In this paper, aiming at providing context-aware and user-specific IoT services, we adopt a device-oriented service model, i.e., each VNF is dedicated to a specific device as a software clone of the user-customized service environment [14], [40]. Moreover, the local data buffer for each VNF instance is modeled as an individual queue. As a result, the ISP can manage network resources in a fine-grained manner and better-personalized service support for highly diverse demands can be achieved. For simplicity, we assume each IoT device is in charge of sensing a specific data type that is different from others [7], [12]. Meanwhile, the generated tasks are computationally intensive and stochastically arrived at, which need to be offloaded to the corresponding VNF for processing. Owing to the convenient virtualization technology and the cheaper storage costs, this simplification can be easily extended to deal with the situation that multiple IoT devices require the service of the same type by initializing a dedicated VNF for each IoT device.

2) Service Migration: For timely queue backlog processing, VNF should be migrated dynamically to the desired edge server endowed with sufficient computing resources. To specify the deployment of VNF, we define a binary variable as the placement indicator, where $x_n^s(t) = 1$ indicates the VNF instance serving IoT device n is placed on edge server s at time slot t, and $x_n^s(t) = 0$ if otherwise. Moreover, there is exactly one instance for each type of VNF to be deployed among all edge servers. Thus, service migration decision x_n^s should satisfy the following constraints

$$\sum_{s \in \mathcal{S}} x_n^s(t) = 1, \forall n \in \mathcal{N}. \tag{1}$$

Denote $f_n^s(t)$ the assigned computing resources to the VNF serving IoT device n. For each edge server, the allocated computing resource to all hosted VNFs can not exceed its maximum CPU frequency F_s, thus we have

$$\sum_{n \in \mathcal{N}} x_n^s(t) f_n^s(t) \leq F_s, \forall s \in \mathcal{S}. \tag{2}$$

3) Dynamic Queue Scheduling: To capture the stochastic and intermittent nature of the data arrivals at VNF queues, we denote the amount of data transmitted by the n^{th} IoT device to its corresponding VNF as $A_n(t)$ (in bits), which is generally different among IoT devices. During each time slot t, denote by $D_n^a(t)$ (in bits) the sensing results at IoT device n. We assume that the arrival $D_n^a(t)$ follows a general i.i.d. distribution, i.e., $\mathbb{E}[D_n^a(t)] = \lambda_n \leq \infty, n = 1, \ldots, N$ [19]. However, the transmitted data size by IoT device is not only related to $D_n^a(t)$ but constrained by the maximum transmission power of the IoT device and the current channel condition. Therefore, the arrival data $A_n(t)$ at the VNF, i.e., the transmitted data volumes from the n^{th} IoT device can be calculated as

$$A_n(t) = \min\left\{D_n^a(t), D_n^o(t)\right\}, \forall n \in \mathcal{N}, \tag{3}$$

where

$$D_n^o(t) = \tau W \left(log_2 \left(1 + \frac{P_n^{\max} h_n(t)}{\sigma^2} \right) \right) \tag{4}$$

is the maximum data size that can be transmitted from IoT device n within time slot t. We assume that all IoT devices adopt frequency division multiple access (FDMA) to offload sensing data, thus the intra-cell and inter-cell interference can be eliminated. W is the allocated channel bandwidth of the associated BS, $P_n^{\max}$ is the maximum transmission power of IoT device n, σ^2 is the noise power, and $h_n(t)$ is channel gain between the IoT device and its local BS at time slot t. To simulate the dynamic communication environment, the channel gains are assumed time-varying, which remain constant within each time slot but vary independently across different slots.

For the VNF that servers n^{th} IoT device, given the current placement strategy and assigned computing resources, the amount of data processed by the VNF within time slot t is denoted as $D_n(t)$ (in bits), which can be expressed by

$$D_n(t) = \sum_{s \in \mathcal{S}} x_n^s(t) \frac{f_n^s(t)\tau}{\phi_n}, \forall n \in \mathcal{N}. \tag{5}$$

ϕ_n is the requested computing cycles per bit for processing the offloaded task of IoT device n.

Denote $Q_n(t)$ as the queue length of the VNF which executes offloaded data from IoT device n. Then, based on the data arrival $A_n(t)$ and executed data size $D_n(t)$ mentioned above, the queue updates at time slot $t+1$ according to the following equation,

$$Q_n(t+1) = \max\left\{Q_n(t) - D_n(t), 0\right\} + A_n(t). \tag{6}$$

Specifically, we assume $Q_n(0) = 0$. For analysis simplicity, we assume the queue capacity is infinite. Nevertheless, we further prove that the queue backlog has an upper bound by adopting our proposed algorithm, which will be illustrated later. Here, we refer to a single queue is strongly stable if the average queue length satisfies

$$\overline{Q}_n = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[Q_n(t)\right] < \infty, \tag{7}$$

where $\mathbb{E}[\cdot]$ means the expectation operation that considers the system state, including the random data arrivals and time-varying channel conditions. Moreover, the system is regarded as strongly stable if all the individual queues of the network satisfy the strongly stable condition [39].

Remark 1: According to Little's Law, the average delay of the task data bit of IoT device n can be written as $\overline{T}_n = \frac{\overline{Q}_n}{\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[A_n(t)]}$, which is directly proportional to the average queue length under a given traffic arrival rate. Therefore, a strongly stable data queue can be translated to a finite processing delay of each task data bit.

4) Energy Consumption: Although dynamical migration of VNFs is necessary to achieve load balancing and ensure network stability, the frequent migration among edge servers causes massive network resource occupation along with expensive operation costs for the ISP. In addition, we propose a simultaneous migration mode, which migrates VNF along with its served queue. This model not only mitigates the system overhead consumed by data synchronization but also reduces migration energy consumption to some extent. In this paper, we aim to reduce service provision costs of the ISP, which consists of transmission energy consumption for migration on backhaul links and computing energy consumption on edge servers.

Specifically, the consumed energy budget for migrating the VNF that serves IoT device n is denoted as

$$E_n^m(t) = \sum_{j\in\mathcal{S}}\sum_{s\in\mathcal{S}} x_n^j(t-1)x_n^s(t)E_n^{j,s}(t), \forall n \in \mathcal{N}, \quad (8)$$

where

$$E_n^{j,s}(t) = \sum_{e_{m,n}\in\mathcal{P}} p\frac{o_n + Q_n(t)}{w_{m,n}}, \forall n \in \mathcal{N} \quad (9)$$

is the energy consumed by transmitting the VNF instance and its queue buffer via the shortest path $\mathcal{P}$ between edge server j and s. p is the transmission power of BS. $w_{m,n}, m, n \in \mathcal{S}$ is the bandwidth of outgoing link $e_{m,n} \in \mathcal{E}$. o_n is the size of the VNF instance that serves IoT device n. From (9), we see that the migration energy budget can be further reduced by effectively controlling the queue length, which is determined by the available resources on the deployed edge server. Therefore, energy saving and load balancing can be achieved by dynamic VNF migration and resource allocation [41]. The computing energy consumption on all edge servers is proportional to their corresponding workload and the square of the CPU speed [27]. Therefore, we define it as

$$E_n^c(t) = \sum_{s\in\mathcal{S}} x_n^s(t)\kappa f_n^s(t)^3\tau, \forall n \in \mathcal{N}, \quad (10)$$

where $\kappa > 0$ is the energy efficiency parameter determined by the structure of edge servers.

The overall energy consumption for the ISP to process the task of IoT device n during time slot t is expressed as

$$E_n(t) = E_n^m(t) + E_n^c(t). \quad (11)$$

For the sake of discussion simplicity, we assume that the edge-edge cooperation is operated within a single ISP, similar to [7]. Considering the profitability attributes in multi-ISP-based MEC systems, the ISPs prefer to use their own deployed edge servers to provide services for the subscribed users [42]. Therefore, the simplification in our paper can be easily extended to the multi-ISP-based MEC systems, by measuring the profits of all ISPs during service provisioning, instead of the energy consumption for the BSs.

B. Problem Formulation

We aim to stabilize the MEC system with extremely bursty data flows in minimal time-averaged operation cost by jointly optimizing the service migration in the network and resource allocation at the edge servers. Specifically, the operating cost of our system is represented by the energy consumption during the computation and migration process. We thus formulate the problem (**P1**) as follows:

$$\min_{\mathbf{x},\mathbf{f}} \quad \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{n=1}^{N} E_n(t) \quad (12)$$

$$\text{s.t.} \quad (1)-(2), (7), \quad (12a)$$

$$f_n^s(t) \geq 0, \quad (12b)$$

$$Q_n(t) \geq D_n(t), \quad (12c)$$

where constraint (12c) is the data causality constraint, indicating that the executed data size on each VNF during time slot t cannot exceed the current queue backlogs. By enforcing (12c), the queue update equation can be simplified as $Q_n(t+1) = Q_n(t) - D_n(t) + A_n(t)$ and $Q_n(t+1) > 0$ always holds for every time slot.

Based on the expression of executed data size (5), constraints (12b) and (12c) can be further integrated as the constraint of computing resources f_n^s, which is derived as

$$0 \leq f_n^s(t) \leq x_n^s(t)\frac{\phi_n Q_n(t)}{\tau}. \quad (13)$$

Problem (**P1**) is a stochastic optimization problem. In general, the derivation of the optimal policy of the long-term optimization problem requires complete offline knowledge to adjust system dynamics, which is extremely hard in realistic situations. To solve the problem in an online manner, the long-term problem needs to be decoupled into a series of real-time optimization problems by carefully handling the long-term objective function and queue stability constraints. However, even though the per-slotted optimization problem is obtained, it is also non-trivial to be solved. Below we prove the NP-hardness of our problem through a reduction from the generalized assignment problem (GAP) by removing time coupling, flexible resource allocation, and queue stability constraints. This reveals that our problem is indeed more complex than the GAP. Therefore, efficient algorithms need to be further designed.

Theorem 1: The optimization problem at each time slot is NP-hard.

Proof: We prove the NP-hardness of the joint optimization problem by a reduction from the well-known GAP already proved to be NP-hard, which contains N jobs and M available machines. Each job i is to be performed on one and only one machine and each machine is constrained by the total available resources b_j. Assigning job i to machine j requires resources a_i^j and consumes costs c_i^j, respectively. GAP is defined as assigning N jobs to M machines at a minimized total cost.

We consider a particular case of our joint optimization problem by assuming that the system stability can always be obtained. In addition, we only focus on solving the service migration subproblem at each time slot. The machine in GAP corresponds to the edge server s with the computing resource constraint F_s. The job in GAP corresponds to the VNF. The assignment process of GAP is equal to placing all VNFs on the optimal edge servers. Given the placement scheme, each VNF consumes energy cost $c_i^j = E_n(x_n^s, f_n^s)$ and requires resources $a_i^j = f_n^s$. The objective of our simplified problem is to minimize the consumed costs, i.e., total energy consumption, subject to the computing resources

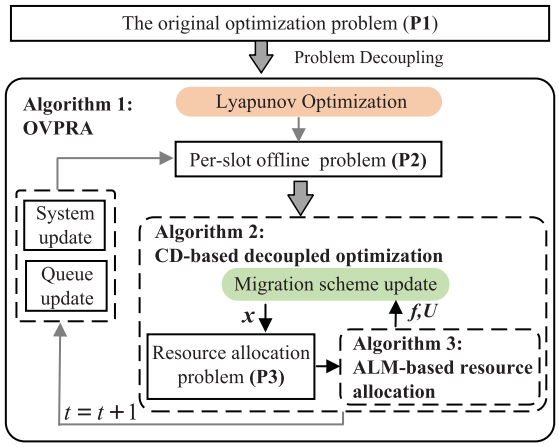

Fig. 2. The proposed optimization framework to solve the problem (**P1**).

on edge servers. It can be seen that our problem can be reduced from GAP in polynomial time, which follows that our problem is NP-hard, q.e.d. □

IV. Online Service Migration and Resource Allocation Algorithm

This section proposes a control framework, namely Online VNF placement and Resource Allocation (OVPRA), for solving the proposed problem in sequential time slots and then proves its performance guarantee on the cost-stability trade-off. The overall solution framework for the problem (**P1**) is depicted in Fig. 2, and the pseudo-code of the OVPRA is illustrated in Algorithm 1. More specifically, we first adopt the Lyapunov optimization theory to reformulate the original problem (**P1**) as a series of offline deterministic problems for the joint optimization of queue stability and service provision costs [39]. Then, in each time slot, we develop a CD-based decoupled optimization framework, as seen in **Algorithm 2**, to solve the per-slot problem (**P2**) according to the current network state, in which the CD algorithm is leveraged to update service migration decisions. Moreover, the ALM-based algorithm illustrated in Algorithm 3 is nested in Algorithm 2 to solve the resource allocation problem (**P3**) with the migration scheme $\mathbf{x}$ in the current iteration and evaluate $\mathbf{x}$ for further update.

A. Problem Decoupling Via Lyapunov Optimization

As a particularly suitable solution to optimize the long-term performance of stochastic networks, Lyapunov optimization has gained sufficient attention for online strategy design [19], [20], [21], [43]. In this subsection, we adopt the Lyapunov optimization framework to transform the time-averaged optimization problem into per-slot deterministic queue stability subproblems without any prior knowledge of channel information and task date arrival.

We first denote the queue backlog set as $\mathbf{Q}(t) = \{Q_n(t)\}_{n=1}^N$, then we adopt the quadratic Lyapunov function to measure the system queue length [39], which is defined as

$$L(\mathbf{Q}(t)) = \frac{1}{2}\sum_{n=1}^{N} Q_n(t)^2. \tag{14}$$

The function is always non-negative and equal to zero if and only if all components are zero. Then, we define the conditional Lyapunov drift over one slot as

$$\Delta L(\mathbf{Q}(t)) = \mathbb{E}\left[L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t))|\mathbf{Q}(t)\right], \tag{15}$$

which is the expected change of the Lyapunov function from one slot to the next with the given queue state. Intuitively, minimizing $\Delta L(\mathbf{Q}(t))$ would push queues to a lower congestion state and system queues are stable when it goes towards a bounded value. Aiming at minimizing the system energy consumption in addition to guaranteeing queue stability, we use a drift-plus-penalty function according to Lyapunov optimization theory [19], [20]. Specifically, at each time slot, a good control strategy can be obtained by greedily minimizing the following drift metric

$$\Delta L(\mathbf{Q}(t)) + V\mathbb{E}\left[E(t)|\mathbf{Q}(t)\right], \tag{16}$$

where $V \geq 0$ is a control parameter to adjust the attention on the queue stability and energy consumption. By pushing the function (16) to its lower bound at each time slot, the system will be sufficiently stable as long as the arrival data rates less than the capacity region boundary of the network [44].

Theorem 2: For arbitrary queue backlogs over all possible control strategies, the drift-plus-penalty function (16) has the following supremum bound at time slot t

$$B + \sum_{n=1}^{N} V\mathbb{E}\left[E_n(t)|\mathbf{Q}(t)\right] + Q_n(t)\mathbb{E}\left[A_n(t) - D_n(t)|\mathbf{Q}(t)\right], \tag{17}$$

where $B = \frac{1}{2}\sum_{n=1}^{N}\left([A_n^{\max}(t)]^2 + [D_n^{\max}(t)]^2\right)$ is a finite constant.

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

After removing the constant that is not concerned with the control decision from (17), the objective function of the per-slot optimization problem can be simplified as

$$\sum_{n=1}^{N} VE_n(t) - Q_n(t)D_n(t). \tag{18}$$

Recalling the definitions of the executed data size of VNF and energy consumption in equations (5) and (11), respectively, we rearrange the problem for a concise form by plugging these equations into (18). We can obtain the following optimization problem ($\mathbf{P2}$):

$$\min_{\mathbf{x},\mathbf{f}} \quad \sum_{n=1}^{N}\sum_{k=1}^{S} x_n^s(t)\left(\rho_n^s(t) + V\tau\kappa f_n^s(t)^3 - Q_n(t)\frac{f_n^s(t)\tau}{\phi_n}\right) \tag{19}$$

$$\text{s.t.} \quad (1)-(2), (13). \tag{19a}$$

$\rho_n^s(t) = V\sum_{j=1}^{S} x_n^j(t-1)E_n^{j,s}(t)$ is a constant since the service migration information at time slot $t-1$ is always known. Therefore, the close-to-optimal control decisions can be obtained by solving the problem ($\mathbf{P2}$), which only needs the current information on channel states and queue backlogs.

Algorithm 1: OVPRA.

Input: The network topology $\mathcal{G}$.

Output: the close-to-optimal service migration and resource allocation decisions during a long period.

1: **Initialization:** $\{Q_n(0)\}_{n=1}^N = 0$
2: **for** *each time slot* $t = 1, 2, \ldots, T$ **do**
3: Observe current system information
4: Solve problem $(\mathbf{P2})$ by adopting Algorithm 2
5: Update queue backlogs by (6)
6: **end for**

Although the time coupling is removed, the problem is still an MINLP problem, which is proven NP-hard and hard to solve in polynomial time [45]. Additionally, due to the coupling constraints (2) and (13) and the non-convex objective function (19), the problem cannot be directly solved. To this end, we propose a decoupled optimization framework with high efficiency and low calculation complexity, which will be introduced in Section IV-B. Consequently, we propose OVPRA to solve the problem (**P1**), as summarized in Algorithm 1. More specifically, OVPRA observes system information from the network topology $\mathcal{G}$ at each time slot t, including $Q_n(t)$ and $E_n(t)$. Then, the close-to-optimal service migration and resource allocation decisions can be obtained by adopting Algorithm 2 to solve the problem $(\mathbf{P2})$. After that, the queue buffers at the next time slot are updated accordingly based on (6).

B. Coordinate Descent Based Decoupled Optimization

At each time slot, the control strategy, including both binary service migration decision and the continuous computing resource allocation should be decided based on the current queue backlogs. Note that the resource allocation problem can be efficiently solved once the service placement is determined. Based on this observation, we propose a decoupled optimization framework by employing the coordinate descent (CD) method to optimize service migration, in which a distributed optimization method is designed to optimize computing resource allocation based on the augmented Lagrangian method (ALM) and primal decomposition. The details are presented in Algorithm 2.

1) Coordinate Descent for Service Migration: At time slot t, given the service migration scheme $\mathbf{x}^t = \{x_n^s\} \in \mathbb{R}^{N\times S}$, the optimal objective function value of $(\mathbf{P2})$ can be obtained by optimizing resource allocation $\mathbf{f}^t$. Specifically, the corresponding algorithm is introduced in Algorithm 3 in the next subsection. Therefore, the cost of carrying out the current placement decision $\mathbf{x}^t$ is given by

$$R(\mathbf{x}^t|\{Q_n(t)\}_{n=1}^N) = \sum_{n=1}^{N}\sum_{s=1}^{S} x_n^s(t)\Big(\rho_n^s(t) + V\tau\kappa\left(f_n^s\right)^3 - Q_n(t)\frac{f_n^s(t)\tau}{\phi_n}\Big). \tag{20}$$

Then, the optimal service migration strategy can be obtained by solving the following problem

$$\min_{\mathbf{x}^t} \quad R(\mathbf{x}^t) \tag{21}$$

$$\text{s.t.} \sum_{s\in\mathcal{S}} x_n^s(t) = 1, \forall n \in \mathcal{N}, \tag{21a}$$

$$x_n^s(t) \in \{0,1\}, \forall n \in \mathcal{N}, \forall s \in \mathcal{S}. \tag{21b}$$

Intuitively, the optimal solution can be obtained by employing the exhaustive search method, which is an exponential enumeration placement scheme. However, this poses high computational complexity especially when the scenario expands. As a popular method for tackling a series of non-convex optimization problems, the main idea of the CD method is to tackle the original problem directly by iteratively optimizing along a single dimension by linear search, while keeping the rest of coordinates fixed [46]. The key advantage of the method lies in the simplicity of each iteration, which is reflected not only in generating the search direction but also in performing the update of the variable. Therefore, it offers straightforward practical implementation and requires less calculation complexity compared to the joint optimization of variables in complex optimization problems.

The detailed execution process is shown in Algorithm 2, which mainly contains two nested loops. Specifically, we initialize a feasible service migration decision as $\mathbf{x}^0$ and denote by $\mathbf{x}^k$ the service migration strategy in the k^{th} iteration. Here, we remove the mark related to the time slot and replace it with the iterations for symbol simplification. At the outer loop (line 4-14 of Algorithm 2), the CD method successively optimizes along the direction of IoT devices to update $\mathbf{x}_{outer}^k$ accordingly. Specifically, $\mathbf{x}_{outer}^k$ represents the local optimal service migration scheme in the k^{th} iteration. Considering that each VNF can only select one of the edge servers for deploying, we alternately update the migrate decision for the VNF that serves IoT device n at the inner loop (line 6-11 of Algorithm 2). Let $U_{n,s}^k$ denote the gain of the swap operation, which is calculated as

$$U_{n,s}^k = R(\mathbf{x}_{outer}^k) - R(\mathbf{x}_{outer}^k(n,s)). \tag{22}$$

$\mathbf{x}_{outer}^k(n,s)$ is the service migration scheme after swapping the migration decision of the VNF serving IoT device n from its current decision to the edge server s, while keeping the decisions of the rest of VNFs fixed.

After the VNF of IoT device n alternatively swaps its selection to all available edge servers, the outer loop variable $\mathbf{x}_{outer}^k$ can be updated accordingly as

$$\mathbf{x}_{outer}^k = \mathbf{x}_{outer}^k(n, s^*), \tag{23}$$

where $s^* = \arg\max_{s=1,\ldots,S}\{U_{n,s}^k > 0\}$ represents the selected edge server with the highest swap gain, if the reward is positive. From (23), the local optimal service migration scheme can be obtained by letting the VNF of n swap its target migration location to the selected edge server s^*. Then, the outer loop will continue along the direction of a new IoT device. The process continues until the gain no longer increases, which means Algorithm 2 converges.

2) Augmented Lagrangian Method Based Computing Resources Allocation Optimization: Given service migration, the migration energy consumption $\sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}} x_n^s\rho_n^s(t)$ in the problem $(\mathbf{P2})$ can be viewed as constant. Therefore, the resource

Algorithm 2: CD-Based Decoupled Optimization.

Input: The current queue backlogs $\{Q_n\}_{n=1}^N$ for all the serving VNFs, network topology $\mathcal{G}$.
Output: The near-optimal service migration decision $(\mathbf{x}^t)^*$.
1: **Initialization:** $k=0$, a feasible service migration decision $\mathbf{x}^0$, $\mathbf{x}_{outer}^0=\mathbf{x}^0$, solution precision ϵ.
2: **repeat**
3: $k=k+1$
4: Outer loop:
5: **for** *each IoT device* $n=1,2,\ldots,N$ **do**
6: Inner loop:
7: **for** *each edge server* $s=1,2,\ldots,S$ **do**
8: Swap the service migration decision from the current location to s
9: Calculate $R(\mathbf{x}_{outer}^k(n,s))$ by adopting Algorithm 3
10: Calculate $U_{n,s}^k$ according to (22)
11: **end for**
12: Update $\mathbf{x}_{outer}^k$ according to (23)
13: **end for**
14: Set $\mathbf{x}^{k-1}=\mathbf{x}_{outer}^k$
15: **until** $c(\mathbf{x}^{k-1})-c(\mathbf{x}^k))\le\epsilon$

allocation optimization problem can be simplified as the following general convex optimization problem ($\mathbf{P3}$):

$$\min_{\mathbf{f}} \quad \sum_{s=1}^{S} p_s \tag{24}$$

$$\text{s.t.} \quad \sum_{n\in\mathcal{N}} x_n^s f_n^s \le F_s, \forall s\in\mathcal{S}, \tag{24a}$$

$$0\le f_n^s \le x_n^s \frac{\phi_n Q_n}{\tau}, \tag{24b}$$

where $p_s=\sum_{n=1}^N x_n^s(\tau V\kappa(f_n^s)^3-Q_n\frac{f_n^s\tau}{\phi_n})$. To tackle the convex problem more effectively, we adopt a distributed method that combines the advantages of regularization terms and primal decomposition to speed up algorithm convergence and optimize in a parallel manner. Therefore, it requires less computation complexity than standard convex optimization tools such as the interior point method. Therefore, the method can be used to satisfy the online-computing requirement for massive IoT devices in large-scale scenarios.

By introducing relaxation variables $\mathbf{y}=\{y_s\}_{s=1}^S\in\mathbb{R}^S$, we first transform the original problem with inequality constraints into the following optimization problem which only contains simple box constraints and equality constraints,

$$\min_{\mathbf{f},\mathbf{y}} \quad \sum_{s=1}^{S} p_s \tag{25}$$

$$\text{s.t.} \quad c_s+y_s=0, \forall s\in\mathcal{S}, \tag{25a}$$

$$y_s\ge 0, \forall s\in\mathcal{S}, \tag{25b}$$

$$0\le f_n^s\le x_n^s\frac{\phi_n Q_n}{\tau}, \forall n\in\mathcal{N}, \forall s\in\mathcal{S}. \tag{25c}$$

Specifically, $c_s=\sum_{n\in\mathcal{N}} x_n^s f_n^s - F_s$.

Then, we construct the augmented Lagrangian function by introducing Lagrangian multipliers $\boldsymbol{\lambda}=\{\lambda_s\}_{s=1}^S\in\mathbb{R}^{S\times 1}$ and appending quadratic penalty functions associated with system requirements, which can be written as

$$L_\delta(\mathbf{f},\mathbf{y},\boldsymbol{\lambda})=\sum_{s=1}^{S} p_s+\sum_{s=1}^{S}\lambda_s(c_s+y_s)+\frac{\delta}{2}\sum_{s=1}^{S}(c_s+y_s)^2, \tag{26}$$

where δ is a positive penalty coefficient. Thus, the corresponding Lagrangian dual function can be written as

$$g(\boldsymbol{\lambda})=\min_{\mathbf{f},\mathbf{y}}\left\{L_\delta(\mathbf{f},\mathbf{y},\boldsymbol{\lambda}), y_s\ge 0, 0\le f_n^s\le x_n^s\frac{\phi_n Q_n}{\tau}\right\}. \tag{27}$$

Moreover, the dual problem is described as

$$\max_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda}), \tag{28}$$

The dual problem can be solved by updating $\{\mathbf{f},\mathbf{y},\boldsymbol{\lambda}\}$ in an iterative manner, the details are presented in Algorithm 3. In particular, with the given variables $\{\mathbf{f}^k,\mathbf{y}^k,\boldsymbol{\lambda}^k\}$ in the k^{th} iteration, the update of all variables in $(k+1)^{th}$ iteration can be sequentially performed as follows.

1) Step 1 (line 3–6 of Algorithm 3): Given $\boldsymbol{\lambda}^k$ and δ^k, we optimize primal variables $\mathbf{f}^{k+1}$, $\mathbf{y}^{k+1}$ by minimizing

$$\left(\mathbf{f}^{k+1},\mathbf{y}^{k+1}\right)=\arg\min_{\mathbf{f},\mathbf{y}} L_{\delta_k}(\mathbf{f},\mathbf{y},\boldsymbol{\lambda}^k), \tag{29}$$

which can be further decomposed into S parallel subproblems based on primal decomposition [47]. The decomposed subproblem is formed as

$$\left(\mathbf{f}_s^{k+1},y_s^{k+1}\right)=\arg\min_{\mathbf{f}_s,y_s} L_{s,\delta_k}(\mathbf{f}_s,y_s), \tag{30}$$

where $L_{s,\delta_k}=p_s+\lambda_s(c_s+y_s)+\frac{\delta_k}{2}(c_s+y_s)^2$ and $\mathbf{f}_s=\{f_n^s\}_{n=1}^N\in\mathbb{R}^N$.

With the given $\mathbf{f}_s^k$, we first minimize L_{s,δ_k} with respect to y_s. According to the optimization theory of convex optimization problem, the global optimal solution can be represented as $y_s^{k+1}=\max\{0,-c_s-\frac{\lambda_s}{\delta_k}\}$. The detail process is omitted for page limits.

By plugging y_s^{k+1} into (30), we then minimize L_{s,δ_k} with respect to $\mathbf{f}_s$, which is formulated as

$$\mathbf{f}_s^{k+1}=\arg\min L_{s,\delta_k}(\mathbf{f}_s), \tag{31}$$

where $L_{s,\delta_k}=p_s+\frac{\delta_k}{2}\max\{0,c_s+\frac{\lambda_s}{\delta_k}\}^2-\frac{(\lambda_s^k)^2}{2\delta_k^2}$. Considering the objective function is continuously differentiable, the gradient method, e.g., the decent gradient method, can thus be adopted to obtain the optimal solution quickly.

After solving all parallel subproblems, the optimal solution of (29) can be obtained as $\mathbf{y}^{k+1},\mathbf{f}^{k+1}$.

2) Step 2 (line 8 of Algorithm 3): Given $\mathbf{y}^{k+1},\mathbf{f}^{k+1}$, the multipliers can be updated according to the KKT conditions of (26) and (30), thus can be expressed as

$$\lambda_s^{k+1}=\left[\lambda_s^k+\delta_k c_s(\mathbf{f}_s)\right]^+, \forall s, \tag{32}$$

where $[\cdot]^+$ is the projection onto the non-negative orthant.

Overall, we repeat *Step 1* and *Step 2* until a specified stopping criterion is satisfied. To further enhance algorithm performance, we utilize solution feasibility to properly adjust the penalty coefficient δ_k and solution accuracy η_k. Specifically, we denote by

Algorithm 3: ALM-Based Computing Resource Allocation.

Input: The current queue backlogs $\{Q_n\}_{n=1}^N$ and migration decision $\{x_n^s\}$ for all the serving VNFs.
Output: The optimal resource allocation solution $\mathbf{f}^*$.

1: **Initialization:** $k=0$, $\mathbf{f}^0$, $\boldsymbol{\lambda}^0$, $0<\alpha\leq 1$, $\mu>1$, $\epsilon>0$, $\eta>0$, $\delta_0>0$, $\epsilon_0=\frac{1}{\delta_0}$, $\eta_0=\frac{1}{\delta_0^\alpha}$.
2: **repeat**
3: **for** *each edge server* $s=1,2,\ldots,S$ **do**
4: Calculating the optimal y_s^{k+1}
5: Solving the subproblem (31) until $\|\nabla_f L_{s,\delta_k}(\mathbf{f}_s, y_s^{k+1}, \lambda_s^k)\|\leq \eta_k$ is met, and then broadcasting the solution $\mathbf{f}_s^{k+1}$
6: **end for**
7: **if** $v_k(\mathbf{f}^{k+1})\leq \epsilon_k$ **then**
8: Update multipliers $\boldsymbol{\lambda}$ according to (32)
9: Update solution accuracy $\epsilon_{k+1}=\frac{\epsilon_k}{\delta_{k+1}}$, $\eta_{k+1}=\frac{\eta_k}{\delta_{k+1}^\alpha}$
10: $\delta_{k+1}=\delta_k$
11: **else**
12: $\boldsymbol{\lambda}^{k+1}=\boldsymbol{\lambda}^k$
13: Update penalty factor $\delta_{k+1}=\mu\delta_k$
14: Adjust solution accuracy $\epsilon_{k+1}=\frac{1}{\delta_{k+1}}$, $\eta_{k+1}=\frac{1}{\delta_{k+1}^\alpha}$
15: **end if**
16: $k=k+1$
17: **until** $v_k(\mathbf{f}^{k+1})\leq\epsilon$ and $\|\nabla_f L_{\delta_k}(\mathbf{f}^k,\lambda_k)\|\leq\eta$

$v_k(\mathbf{f}^{k+1})=\sqrt{\sum_{s\in\mathcal{S}}\left(c_s(\mathbf{f}_s^{k+1})+y_s^{k+1}\right)^2}$ the solution feasibility at each iteration. ϵ_k represents the feasibility tolerance, which is a sufficiently small positive value. When $v_k(\mathbf{f}^{k+1})\leq\epsilon_k$, the multipliers should be updated accordingly and accuracy parameters $\epsilon_{k+1},\eta_{k+1}$ should be improved to obtain a more accurate solution at the next iteration. If otherwise, the multipliers remain unchanged and the penalty factor δ_{k+1} should be properly scaled up to reduce $v_k(\mathbf{f}^{k+1})$ possibly.

C. Performance Analysis

The execution of OVPRA mainly lies in the per-slot offline optimization for the problem **(P2)** (line 4in Algorithm 1). Hence, in this section, we focus on analyzing the Algorithm 2 in convergence performance and computational complexity. Moreover, we analyze the performance of OVPRA based on the drift-plus-penalty algorithm in joint stability and average energy cost minimization.

1) Convergence Performance Analysis: We first provide the convergence analysis of the Algorithm 3 (line 9 in Algorithm 2), which is invoked in Algorithm 2 at each iteration. Then, we provide the convergence analysis of the Algorithm 2.

Proposition 1: Our proposed Algorithm 3 is guaranteed to converge within a limited number of iterations.

Proof: Since the duality gap of the problem $(\mathbf{P3})$ is zero and a unique optimal solution (29) exists, it can be concluded that the primal variables $\mathbf{f}^*$ can converge to the optimal value eventually [48]. Moreover, according to the Lagrangian multipliers update rule (32), the optimal dual point can also be obtained when the multipliers converge [47]. Therefore, Algorithm 3 can be guaranteed to converge to the global optimal solution while the algorithm reaches the termination. □

Proposition 2: Our proposed Algorithm 2 can converge within a limited number of iterations.

Proof: During the optimization procedure of Algorithm 2, the nested resource allocation Algorithm 3 at each iteration is proved to be convergence. Moreover, the objective function value $R(\mathbf{x}^t)$ is non-increasing as the iterations proceed. As for the problem $(\mathbf{P2})$, the optimal policy exists in general, which corresponds to a bounded minimum of the objective function. Therefore, it is guaranteed to converge and obtain a near-optimal solution within a limited number of iterations. □

2) Computational Complexity Analysis: Here, we first analyze the complexity of Algorithm 3 and then present the overall computational complexity of the Algorithm 2.

Proposition 3: The total complexity of Algorithm 3 is approximated by $O(SN\frac{1}{\eta}))$.

Proof: In Algorithm 3, the computational complexity is mainly related to the update of primal and dual variables. By leveraging primal decomposition, the complexity of solving each subproblem in (30) does not scale with S. In particular, we adopt the gradient method to optimize $\mathbf{f}_s^{k+1}$. The gradient calculation complexity is $O(N)$, which is related to the variable dimension N. In addition, for strongly convex functions, optimization theory informs us that the gradient descent method requires $O(\frac{1}{\eta})$ iterations to obtain η solution accuracy. Therefore, the overall computational complexity for optimizing $\mathbf{f}_s^{k+1}$ is calculated as $O(N\frac{1}{\eta})$. Additionally, the computational complexity for optimizing y_s^{k+1} is $O(1)$ since the optimal value can be obtained by the closed-form solution. Therefore, the overall computational complexity of Step 1 is $O(SN\frac{1}{\eta})$. Moreover, the computational complexity of dual variables update is $O(S)$. In summary, the total complexity of Algorithm 3 can be approximated by $O(SN\frac{1}{\eta}))$. □

Proposition 4: The overall complexity of Algorithm 2 is approximated by $O(IS^2N^2\frac{1}{\eta}))$.

Proof: In Algorithm 2, the computational complexity mainly lies in the double loops (lines 4–13 in Algorithm 2), in which the Algorithm 3 is performed. Therefore, the complexity at each iteration can be approximated by $O(S^2N^2\frac{1}{\eta}))$. Leaving I denote the number of iterations required to solve $(\mathbf{P2})$, the overall computational complexity can thus be approximated as $O(IS^2N^2\frac{1}{\eta}))$. □

Based on computational complexity analysis, we conclude that the proposed CD-based decoupled optimization framework requires less computational complexity compared to the exhaustive search method with a complexity of $O(S^N\frac{1}{\eta})$.

3) Analysis of the Drift-Plus-Penalty-Based OVPRA: This section concludes the theoretical analysis of the proposed OVPRA framework on the performance of average energy consumption and system stability.

Definition 1: The network stability region Λ is the closure of the set of all arrival rate vectors $\{\lambda_n|n\in\mathcal{N}\}$ that can be stably supported by all possible control policies [49]. Mathematically, it can be expressed as

$$\Lambda=\bigcup_{\Omega\in G}\Lambda_\Omega,$$

where $\Lambda_\Omega = \{\lambda_n | n \in \mathcal{N}\}$ is the stability region of a stabilizing control policy Ω, and G is all possible strategies for choosing the control variables to affect scheduling and resource allocation.

Theorem 3: Let $\boldsymbol{\lambda} = \{\lambda_n | n \in \mathcal{N}\}$ represent the arrival rate vector. Assuming that $\boldsymbol{\lambda}$ is interior to the network stability region Λ, for which there exists a sufficiently small value $\epsilon > 0$ such that $\boldsymbol{\lambda} + \boldsymbol{\epsilon} \in \Lambda$, where $\boldsymbol{\epsilon}$ is a vector with entries $\epsilon_n = \epsilon$. Then, the following conditions will always hold by adopting the OVPRA for any value of V:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \mathbb{E}\left[E(t)\right] \leq \frac{B}{V} + e^*, \tag{33}$$

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} \sum_{n=1}^{N} \mathbb{E}\left[Q_n(t)\right] \leq \frac{B + V\left(e^* - \overline{e}\right)}{\epsilon}. \tag{34}$$

Λ is the network stability region. e^* is the minimum time-averaged energy consumption that makes all queues rate stable with the given data arrival rates $\{\lambda_n\}_{n=1}^N$ of all IoT devices. $\overline{e}$ is the time-averaged energy consumption, and $\overline{e} \geq e^*$. B is a constant that has denoted in (17).

Proof: Please refer to Appendix B. □

Theorem 3 illustrates the performance bounds of our proposed OVPRA framework on time-averaged system energy consumption and queue backlogs. It can be seen that the OVPRA is parameterized by a constant $V \geq 0$ and results in the $[O(1/V), O(V)]$ cost-stability trade-off. That is to say, choosing a large value of V can push average energy consumption arbitrarily close to optimal. However, this comes with a tradeoff in the increasing queue backlogs that have a linear relationship with V. Therefore, the control parameter V can be appropriately adjusted as desired to yield time-averaged energy consumption within $O(1/V)$ from the minimum possible time-averaged energy consumption required for queue stability. Meanwhile, in practice, V can be chosen according to the requirements and characteristics of the ISP and IoT devices in the MEC network.

V. Numerical Results

In this section, we present simulations to verify the effectiveness of our algorithms and analysis proposed in the previous sections. We consider an MEC network with four edge servers within a square area of 800 m × 800 m. Several IoT devices are randomly deployed in the scenario and generate different types of compute-intensive data, such as sensor data, environment data, wearable health data, and so forth. Each IoT device continuously connects to an always-on exclusive VNF to achieve edge-assisted sensing. The generating sensing data on IoT devices are set to follow a certain fixed statistical distribution, which is assumed as the exponential distribution with equal average rate, i.e., $\mathbb{E}[D_n^a(t)] = \lambda_n, n = 1, \ldots, N$. To simulate the time-varying environment, we assume the average channel gain that follows the free-space path loss model as $\bar{h}_n = A_d(\frac{3\times10^8}{4\pi f_c d_n})^{d_e}, n = 1, \ldots, N$, where A_d is the antenna gain, f_c is the carrier frequency, d_e is the path loss exponent, and d_n is the distance between the IoT device n and its associated BS [19], [50]. The time-varying wireless channel gain for each IoT device is $h_n(t) = \beta(t)\bar{h}_n$, where $\beta(t)$ is the independent random channel fading factor, that obeys Rayleigh distribution. In addition, the default maximum computing resources F_s for all edge servers are assumed equal. Unless otherwise specified, we set $V = 20$, $N = 10$, $\lambda_n = 2$ Mbps, $\kappa = 10^{-26}$. The other key system parameters are listed in Table III.

TABLE III
Simulation Parameters

Parameters	Value
Time slot interval t	2 s
The number of time slots T	2000
Default computing capacity threshold F_s	0.3 GHz
Max IoT device transmit power $P_n^{\max}$	0.1 W
Process density φ_n	100cycles/bit
Wireless bandwidth W	2 MHz
Noise power θ	10^{-9} W
Path loss exponent k	3
Backhaul link bandwidth $w_{i,j}, i, j \in \mathcal{S}$	100Mbps [11]

For performance evaluation, we consider the following benchmarks.

Random Placement (RP): At each time slot, the placement of all VNFs is randomly determined and the resource allocation is obtained by adopting Algorithm 3 with the given VNF placement.

No migration (NM): During a long period, all VNFs always retain their initial placement policy, which means that VNF migration will not be triggered no matter how poor the current communication quality is. This scheme pays more attention to cost performance. Therefore, it provides a lower bound of energy consumption.

Queue length minimization optimization (QueueOpt): At each time slot, QueueOpt only focuses on minimizing the queue backlogs for all VNFs but neglects the energy consumption of the ISP. Therefore, it can provide a loose lower bound of queue backlogs.

Service Placement via Randomized Rounding (SPR2) [38]: By leveraging the randomized rounding technique, the integer service migration decision can be obtained by rounding the solution from the linear relaxation of the problem (**P1**).

Queue-aware dynamic placement of virtual network functions (QNPVNF) [24]: The service placement decision is obtained by leveraging the Lyapunov optimization technique and a genetic algorithm-based heuristic placement algorithm.

Fig. 3 illustrates the convergence of **Algorithm 2**. For a comprehensive evaluation, we randomly generate 100-time slots with varying system parameters, including the number of IoT devices and data arrival rates. Then, we run **Algorithm 2** to obtain the objective function value for each iteration. All points in Fig. 3 are the average of 100-time slots. It can be seen that the algorithm achieves convergence to the optimal value with no more than 4 iterations even with the increasing IoT devices and a larger data arrival rate. Besides, one prominent observation is that the growth of data arrivals has a greater impact on the overall system performance than the number of IoT devices. The results demonstrate that our algorithm can achieve superior performance in the scenario with relatively massive IoT devices, provided that data arrival rates are within the network's stability region.

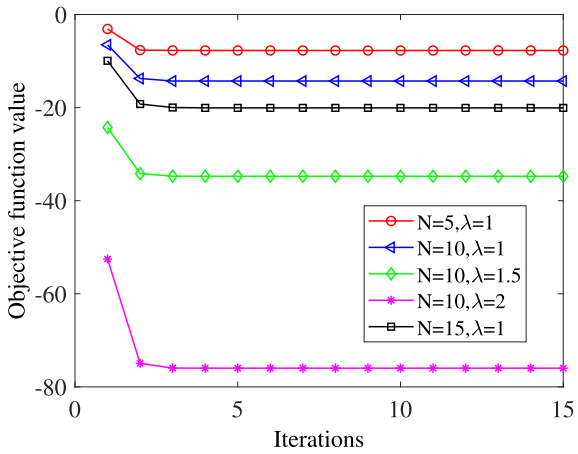

Fig. 3. Convergence performance of Algorithm 2.

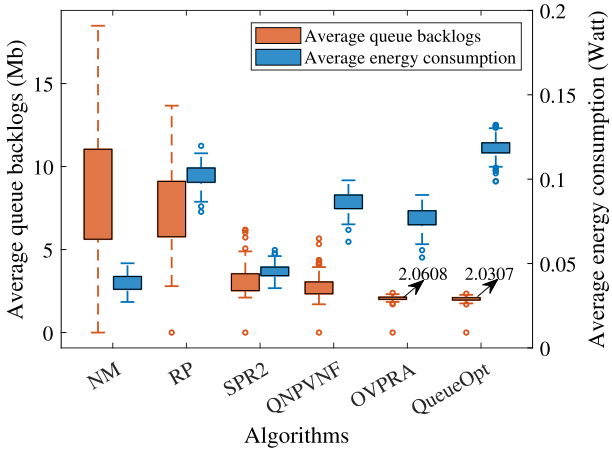

Fig. 4. Performance comparison of different algorithms over 2000 time slots.

Fig. 4 leverages boxplot to illustrate the distribution characteristics of average queue backlogs and energy consumption for different algorithms. Specifically, we consider the random data arrival and channel gain in 2000 time slots and each box represents the distribution of 2000 values. By comparing the average queue backlogs of all schemes, we observe that the NM and RP schemes exhibit a relatively large interquartile range (IQR), which reflects a significant variability and unstable network state during a long period. In addition, although the median values of SRP2 and QNPVNF are maintained at a lower level, several data points significantly deviate from the rest of the dataset, especially under the SPR2 scheme. Therefore, the achieved performance is suboptimal. Conversely, the QueueOpt scheme can always maintain the best performance as expected, and the proposed OVPRA achieves nearly optimal performance. This demonstrates the effectiveness of the OVPRA in stabilizing queue backlogs over the long run. This also testifies that a proper service migration policy contributes to the load balancing on edge servers.

Next, by comparing the average energy consumption of all schemes in Fig. 4, we observe that the NM scheme can always obtain the lowest energy consumption since it always keeps the initial placement policy unchanged even though the edge server cannot bear the hosted queues. Accordingly, QueueOpt does not consider energy consumption during VNF migration and resource allocation. As a result, its average energy consumption is the largest, as expected. Furthermore, we also notice that the proposed OVPRA can achieve a better performance than other benchmarks except SRP2, which verifies its efficiency in reducing operating costs. Although SRP2 incurs lower service provision costs, it cannot guarantee the QoS due to the inability to ensure network stability. In summary, compared with the heuristic algorithm QNPVNF and approximate algorithm SPR2, OVPRA can attain the best performance in terms of network stability, while consuming relatively less energy.

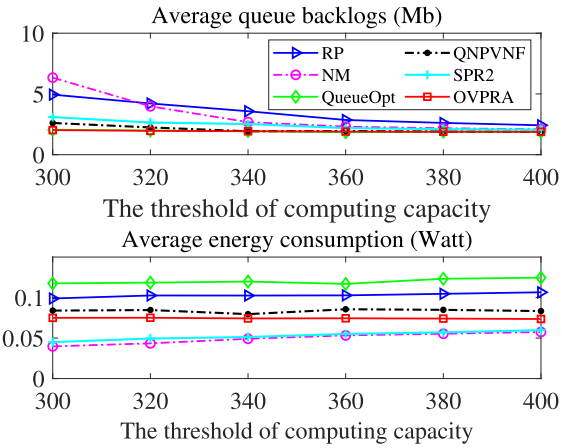

Fig. 5. Performance comparisons versus the threshold of computing capacity.

Fig. 5 illustrates the impact of computing resource constraints of edge servers on the performance of long-term average queue backlogs and energy consumption. At first, we observe that the average queue backlogs decrease as the threshold rises, which can be explained by the fact that most VNF queues possess sufficient computing resources to process the arrival data promptly. Meanwhile, network stability can always be obtained no matter which scheme is adopted, when the provided computing resources are saturated, i.e., $F_s > 380$. Furthermore, the achieved performance of our proposed OVPRA on queue stability is gradually approaching the QueueOpt, even with a stringent computing resources constraint.

Then, we observe that the achieved energy consumption under all schemes increases slightly as the threshold rises and achieves convergence eventually. The reason mainly lies in that VNF migration will be frequently triggered to strive for more computing resources for processing the cached tasks when the computing resources are exiguous. In contrast, in scenarios with sufficient resources, energy consumption is mainly caused by task processing, which is significantly lower than the energy consumed during the simultaneous migration process. Besides, aiming at minimizing the system energy budget, the additional resources are unnecessary for processing the offloaded computing tasks once the network stability is achieved. Therefore, the consumed energy tends to remain unchanged as F_s rises to a certain extent.

Fig. 6 illustrates the impact of data arrival rates on the performance of long-term average queue length and energy

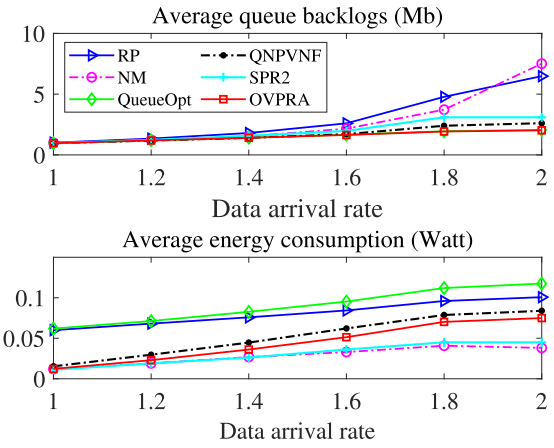

Fig. 6. Performance comparisons versus different arrival data rates.

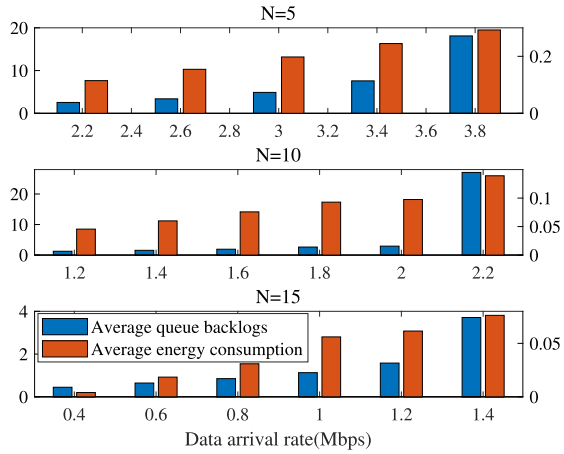

Fig. 7. Performance of OVPRA within stability region versus the number of IoT devices.

consumption. It is obvious to see that the achieved performance of the OVPRA in terms of queue stability approaches that of QueueOpt. Moreover, as the arrival rate increases, the average data queue backlogs under all schemes increase. Notably, the OVPRA exhibits the least amount of curve fluctuation among them, demonstrating its robustness under conditions with heavier computation workloads. In contrast, the queue backlogs of the RP and NM increase sharply when $\lambda \geq 1.6$, and all queues even cannot be stabilized when $\lambda \geq 2$. Therefore, we do not include the situation in Fig. 6.

Then, we see that the average energy consumption under all schemes increases as the arrival rate rises. This can be explained that the VNF migration policy plays a significant role in addressing spatially uneven traffic distribution, aiming at seeking more available computing resources. However, the rate of growth decreases when $\lambda \geq 1.8$. This is because the data arrival rates are gradually approaching the computation capacity. When the available computing resources in the system become extremely scarce, the consumed energy in the data processing phase will no longer increase due to the full utilization of edge computing resources. Meanwhile, since the dynamic VNF migration scheme cannot yield significant performance improvements in terms of network stability, it will not be frequently triggered for cost efficiency. Nevertheless, our proposed OVPRA can achieve a better trade-off between queue stability and energy consumption over a long period, benefiting from the Lyapunov optimization theory and CD-based decoupled optimization framework.

Fig. 7 illustrates the performance of the proposed OVPRA under different numbers of IoT devices. Each subplot reflects the average queue backlogs and energy consumption within the stability region. Obviously, due to the limited computing resources, the upper bound of the stable capacity region descends as the number of IoT devices increases. Specifically, we observe that OVPRA can maintain stable queues for $\lambda \leq 3.8$ when $N = 5$, $\lambda \leq 2.2$ when $N = 10$, and $\lambda \leq 1.4$ when $N = 15$. Meanwhile, for a specific data arrival rate λ, the achieved performance degrades as N increases because of the heavier computation workload in the system. For instance, with $\lambda = 1.4$, the average queue backlogs and energy consumption for $N = 15$ increase

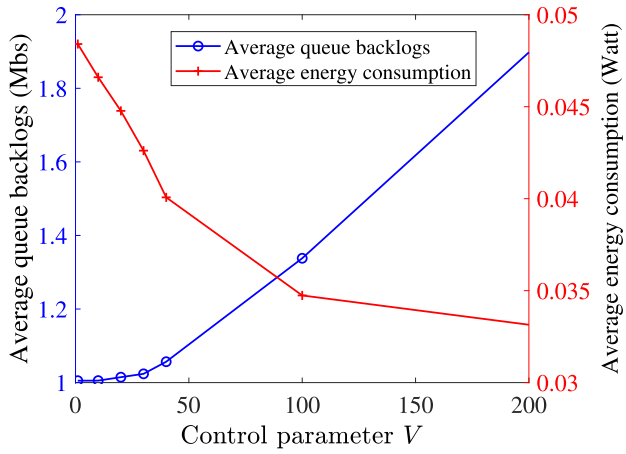

Fig. 8. Performance of OVPRA versus control parameter V.

by nearly 138% and 27%, respectively, compared to $N = 10$. Meanwhile, we observe that the average queue backlogs and energy costs increase as the data arrival rate rises, which also has been demonstrated in Fig. 6.

Fig. 8 illustrates the performance of the OVPRA on the trade-off between average queue backlogs and energy consumption under different control parameters V, where V varies between [10,200]. Obviously, as V increases, the average energy consumption decreases while the average queue backlogs increase, which verifies the rationality of our analysis in Theorem 3. The results can be intuitively explained that a larger V makes the system pay more attention to energy consumption, which drives the OVPRA to curb energy consumption cost-efficiently by reducing the probability of VNF migration and resource allocation of edge servers. Therefore, the OVPRA can achieve the optimal trade-off between queue stability and energy consumption by adjusting the control parameter V to a proper value.

Generally, the ISP always aims to achieve significant energy savings, and the queue backlog also needs to be controlled without a substantial increase. Therefore, the optimal range of

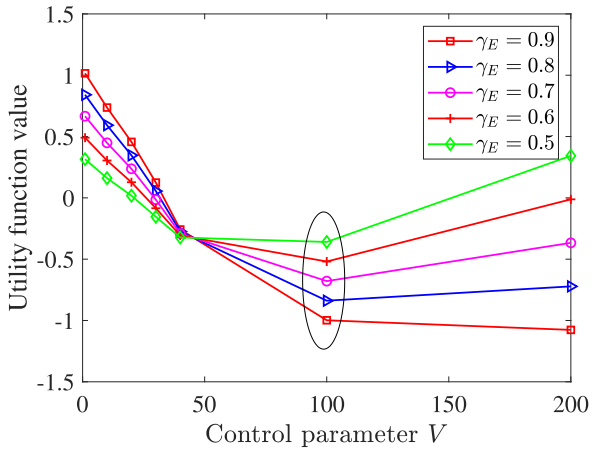

Fig. 9. Utility function value versus control parameter V.

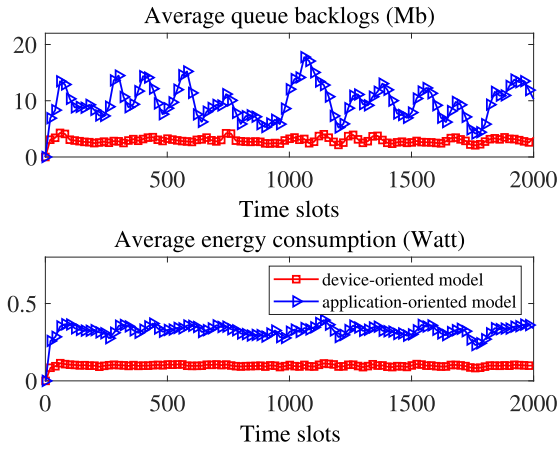

Fig. 10. Performance of OVPRA versus different service models.

V needs to be further analyzed and determined to strike a nice balance between the average energy and queue backlogs. From (34) and (33) in Theorem 3, the average queue length is proportional to V, while the average energy consumption is inversely proportional to V. Therefore, with larger values of V, the energy savings will gradually diminish, while the queue backlogs still exhibit a significant increase. Additionally, when V is extremely small, all queues are generally stabilized and significant energy savings can be achieved, while the consumed energy may exceed the ISP's budget. Intuitively, the optimal range of V should strive to be a median value. To further determine the optimal value of V, a utility function in a weighted sum form can be invoked to evaluate whether the achieved balance is good or not, which can be described as $\min \gamma_e \tilde{E} + (1-\gamma_e)\tilde{Q}$. $\tilde{E}$ and $\tilde{Q}$ are the normalized values of the average energy consumption and the average queue backlogs, respectively, to realize a unified scale of the different matrices. Fig. 9 plots the utility function value under different weight coefficients of γ_e, from which we can observe that the utility function values achieve the minimum when V is set to 100. Therefore, it can be concluded that the optimal range of V in our simulated scenario is approximately 100, which facilitates significant energy savings while controlling the queue backlogs increase slightly.

Fig. 10 plots the performance of our proposed OVPRA in different service models. Here, we assume 10 IoT devices request 5 types of services. In contrast to the device-oriented model, the application-oriented service model assumes that each VNF is dedicated to a specific application, as investigated in [8], [27]. In such scenarios, requests specifying the same application can share the queue buffer and the allocated resources on the same VNF. For simplicity, we set only one application-oriented VNF to be deployed for each service. As illustrated in Fig. 10, the device-oriented service model has superior performance on average energy consumption and queue backlogs. This can be explained by the fact that sharing computing resources and the queue buffer on the application-oriented VNF leads to coarse-grained network management. To be more specific, once one of the served IoT devices emerges bursty data transmission, queue instability on the application-oriented VNF may occur. In order to maintain stability, migration of both VNF and the pending date of all served IoT devices will be frequently triggered. As a result, the increase in migration frequency and data volume causes extensive bandwidth resource wastage, and more migration energy will be consumed subsequently. Besides, due to the sharing of computing resources, more available resources are required for the application-oriented VNF to maintain queue stability. This poses a significant challenge in finding an optimal solution with better computing capacity and less migration cost, thus causing a degraded system performance. Conversely, migration of device-oriented VNF is only influenced by the traffic patterns of its specific IoT device, which enables network management in a user-specific and fine-grained manner.

VI. Conclusion

We investigated a queue-aware service migration and resource allocation problem in resource-limited MEC networks with considering stochastic data arrivals and channel conditions. We first formulated the problem as a long-term MINLP optimization problem to minimize the time-averaged energy cost for the ISP under the constraints of computing resources of edge servers and network stability. However, the main challenge in designing an online control policy that dynamically places VNFs among the edge servers and finds the optimal resource assignment scheme for all hosted VNFs lies in the requirement for accurate prediction of future network information. To this end, we proposed an online optimization framework called OVPRA, which leverages Lyapunov optimization theory to decouple the original problem as a series of per-slot optimization problems. We then proposed a coordinate descent-based decoupled optimization framework with low complexity and fast convergence to efficiently deal with offline problems. We demonstrated the $[\frac{1}{V}, V]$ balance between the energy budget and queue backlog, as well as the convergence of OVPRA, in both theoretical proof and simulations. Moreover, simulation results revealed that OVPRA could achieve energy consumption reduction and network stability simultaneously effectively and robustly.

APPENDIX A
PROOF OF THE THEOREM 2

By plugging the queue updates equation (6) and Lyapunov function (14) into (15), we can drive as

$$\begin{aligned}
&L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t)) \\
&= \frac{1}{2}\sum_{n=1}^{N}\left[Q_n(t+1)^2 - Q_n^2(t)\right] \\
&= \frac{1}{2}\sum_{n=1}^{N}\left[\left(Q_n(t) - D_n(t) + A_n(t)\right)^2 - Q_n^2(t)\right] \\
&= \frac{1}{2}\sum_{n=1}^{N}\left(A_n(t) - D_n(t)\right)^2 + \sum_{n=1}^{N} Q_n(t)\left(A_n(t) - D_n(t)\right) \\
&\le \frac{1}{2}\sum_{n=1}^{N}\left(A_n(t)\right)^2 + \left(D_n(t)\right)^2 + \sum_{n=1}^{N} Q_n(t)\left(A_n(t) - D_n(t)\right) \\
&\le B + \sum_{n=1}^{N} Q_n(t)\left(A_n(t) - D_n(t)\right), \qquad (35)
\end{aligned}$$

where $B = \frac{1}{2}\sum_{n=1}^{N}\left([A_n^{\max}(t)]^2 + [D_n^{\max}(t)]^2\right)$. Specifically, by assuming $D_n^a(t) < D_{\max}^a$, we have $A_n^{\max}(t) = \min\{D_{\max}^a, D_{\max}^o(t)\}$, where $D_{\max}^o(t)$ is determined by the maximum transmission power and channel gain.

Taking conditional expectations of the above equation and adding $V\mathbb{E}[E(t)|\mathbf{Q}(t)]$ to both sides of (16), we thus have

$$\begin{aligned}
\Delta L(\mathbf{Q}(t)) + V\mathbb{E}[E(t)|\mathbf{Q}(t)] \le B + \sum_{n=1}^{N} V\mathbb{E}[E(t)|\mathbf{Q}(t)] \\
+ Q_n(t)\mathbb{E}[A_n(t) - D_n(t)]|\mathbf{Q}(t)]. \qquad (36)
\end{aligned}$$

APPENDIX B
PROOF OF THE THEOREM 3

According to the theorem in [44], the optimal policy exists if the arrival rates of all IoT devices are within the stability capacity region Λ. In particular, if there exists $\epsilon > 0$ that satisfies $\sum_{n=1}^{N}\lambda_n + \epsilon < \Lambda$, we have the the following formulas

$$\mathbb{E}\left\{E(t)\right\} = e^*, \qquad (37)$$

$$\mathbb{E}\left\{A_n(t)\right\} \le \mathbb{E}\left\{D_n(t)\right\} - \epsilon, \qquad (38)$$

where e^* is the average energy consumption under the optimal policy. Plugging the above equations to (17), the upper bound of the drift plus penalty function can be rewritten as

$$\Delta L(\mathbf{Q}(t)) + V\mathbb{E}\left[E(t)|\mathbf{Q}(t)\right] \le B + Ve^* - \epsilon\sum_{n=1}^{N} Q_n(t). \qquad (39)$$

By taking expectations of both sides on the (16) and summing over time slots, the equation can be derived based on the law of iterated expectations as

$$\begin{aligned}
\mathbb{E}\left[L(\mathbf{Q}(T))\right] - \mathbb{E}\left[L(\mathbf{Q}(0))\right] + \sum_{t=0}^{T} V\mathbb{E}\left[E(t)\right] \\
\le T(B + Ve^*) - \epsilon\sum_{t=0}^{T-1}\sum_{n=1}^{N}\mathbb{E}\left[Q_n(t)\right]. \qquad (40)
\end{aligned}$$

Generally, we assume that $Q_n(0) = 0$, we thus have $\mathbb{E}[L(\mathbf{Q}(T))] - \mathbb{E}[L(\mathbf{Q}(0))] \ge 0$. In addition, since $\epsilon > 0$, there holds that $\epsilon\sum_{t=0}^{T-1}\sum_{n=1}^{N}\mathbb{E}[Q_n(t)] \ge 0$. Thus, the inequality is still valid after removing the above non-negative parts, which can be expressed by

$$\sum_{t=0}^{T-1} V\mathbb{E}\left[E(t)\right] \le T(B + Ve^*). \qquad (41)$$

By dividing (41) by VT and letting $T \to \infty$, we have

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[E(t)\right] \le \frac{B}{V} + e^*. \qquad (42)$$

Similarly, based on (40), we also have

$$\epsilon\sum_{t=0}^{T-1}\sum_{n=1}^{N}\mathbb{E}[Q_n(t)] \le T(B + Ve^*) - \sum_{t=0}^{T} V\mathbb{E}\left[E(t)\right]. \qquad (43)$$

Let $T \to \infty$ and divide ϵT at both sides of (43), we also obtain

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T}\sum_{n=1}^{N}\mathbb{E}\left[Q_n(t)\right] \le \frac{B + V\left(e^* - \overline{e}\right)}{\epsilon}, \qquad (44)$$

where $\overline{e} = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[E(t)]$ is the time-averaged energy consumption.

REFERENCES

[1] W. Rafique, L. Qi, I. Yaqoob, M. Imran, R. U. Rasool, and W. Dou, "Complementing IoT services through software defined networking and edge computing: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1761–1804, Thirdquarter 2020.

[2] J. Pan and J. McElhannon, "Future edge cloud and edge computing for Internet of Things applications," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449, Feb. 2018.

[3] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, "Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 38–67, Firstquarter 2020.

[4] 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, Firstquarter 2016.

[5] R. Lin, L. He, S. Luo, and M. Zukerman, "Energy-aware service function chaining embedding in NFV networks," *IEEE Trans. Serv. Comput.*, vol. 16, no. 2, pp. 1158–1171, Mar./Apr. 2023.

[6] Y. Xiao et al., "NFVdeep: Adaptive online service function chain deployment with deep reinforcement learning," in *Proc. IEEE/ACM 27th Int. Symp. Qual. Service*, 2019, pp. 1–10.

[7] Z. Xu, W. Gong, Q. Xia, W. Liang, O. F. Rana, and G. Wu, "NFV-enabled IoT service provisioning in mobile edge clouds," *IEEE Trans. Mobile Comput.*, vol. 20, no. 5, pp. 1892–1906, May 2021.

[8] 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.

[9] Z. Ning et al., "Distributed and dynamic service placement in pervasive edge computing networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 6, pp. 1277–1292, Jun. 2021.

[10] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 3, pp. 939–951, Mar. 2021.

[11] Q. Yuan, J. Li, H. Zhou, T. Lin, G. Luo, and X. Shen, "A joint service migration and mobility optimization approach for vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 9041–9052, Aug. 2020.

[12] Y. Ma, W. Liang, J. Li, X. Jia, and S. Guo, "Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 1, pp. 196–210, Jan. 2022.

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

[14] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.

[15] T. Liu, S. Ni, X. Li, Y. Zhu, L. Kong, and Y. Yang, "Deep reinforcement learning based approach for online service placement and computation resource allocation in edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 3870–3881, Jul. 2023.

[16] C. Dong, Y. Sun, M. Shafiq, N. Hu, Y. Liu, and Z. Tian, "Optimizing mobility-aware task offloading in smart healthcare for internet of medical things through multiagent reinforcement learning," *IEEE Internet Things J.*, vol. 11, no. 8, pp. 13677–13691, Apr. 2024.

[17] M. Chen, Y. Miao, Y. Hao, and K. Hwang, "Narrow band Internet of Things," *IEEE Access*, vol. 5, pp. 20557–20577, 2017.

[18] Z. Zhuang, J. Wang, Q. Qi, J. Liao, and Z. Han, "Adaptive and robust network routing based on deep reinforcement learning with Lyapunov optimization," in *Proc. IEEE/ACM 28th Int. Symp. Qual. Service*, 2020, pp. 1–10.

[19] S. Bi, L. Huang, H. Wang, and Y.-J. A. Zhang, "Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7519–7537, Nov. 2021.

[20] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. Shen, "Energy efficient dynamic offloading in mobile edge computing for Internet of Things," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 1050–1060, Jul.–Sep. 2021.

[21] Z. Yang, S. Bi, and Y.-J. A. Zhang, "Dynamic offloading and trajectory control for UAV-enabled mobile edge computing system with energy harvesting devices," *IEEE Trans. Wireless Commun.*, vol. 21, no. 12, pp. 10515–10528, Dec. 2022.

[22] T. Wang, Y. Zhang, N. N. Xiong, S. Wan, S. Shen, and S. Huang, "An effective edge-intelligent service placement technology for 5G-and-beyond industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 18, no. 6, pp. 4148–4157, Jun. 2022.

[23] L. Chen et al., "IoT microservice deployment in edge-cloud hybrid environment using reinforcement learning," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12610–12622, Aug. 2021.

[24] L. Tang, H. Yang, R. Ma, L. Hu, W. Wang, and Q. Chen, "Queue-aware dynamic placement of virtual network functions in 5G access network," *IEEE Access*, vol. 6, pp. 44291–44305, 2018.

[25] A. Medeiros, A. D. Maio, T. Braun, and A. Neto, "TENET: Adaptive service chain orchestrator for MEC-enabled low-latency 6DoF virtual reality," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 2, pp. 1894–1911, Apr. 2024.

[26] W. Chu, P. Yu, Z. Yu, J. C. Lui, and Y. Lin, "Online optimal service selection, resource allocation and task offloading for multi-access edge computing: A utility-based approach," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 4150–4167, Jul. 2023.

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

[28] C. Feng, Q. Yang, T. Q. Quek, W. Wu, and K. Guo, "Spatially-temporally collaborative service placement and task scheduling in MEC networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 12, pp. 16650–16666, Dec. 2023.

[29] Y. Ma et al., "A performance and reliability-guaranteed predictive approach to service migration path selection in mobile computing," *IEEE Internet Things J.*, vol. 10, no. 20, pp. 17977–17987, Oct. 2023.

[30] N. Kiran, X. Liu, S. Wang, and C. Yin, "VNF placement and resource allocation in SDN/NFV-enabled MEC networks," in *Proc. 2020 IEEE Wireless Commun. Netw. Conf. Workshops*, 2020, pp. 1–6.

[31] W. Liu, H. Zhang, H. Ding, Z. Yu, and D. Yuan, "QoE-aware collaborative edge caching and computing for adaptive video streaming," *IEEE Trans. Wireless Commun.*, vol. 23, no. 6, pp. 6453–6466, Jun. 2024.

[32] X. Gao, R. Liu, A. Kaushik, and H. Zhang, "Dynamic resource allocation for virtual network function placement in satellite edge clouds," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2252–2265, Jul./Aug. 2022.

[33] J. Du et al., "MADDPG-based joint service placement and task offloading in MEC empowered air–ground integrated networks," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 10600–10615, Mar. 2024.

[34] S. S. Shinde and D. Tarchi, "Hierarchical reinforcement learning for multi-layer multi-service non-terrestrial vehicular edge computing," *IEEE Trans. Mach. Learn. Commun. Netw.*, vol. 2, pp. 1045–1061, 2024.

[35] S. S. Shinde and D. Tarchi, "Multi-time-scale Markov decision process for joint service placement, network selection, and computation offloading in aerial IoV scenarios," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 6, pp. 5364–5379, Nov./Dec. 2024.

[36] W. Fan et al., "Collaborative service placement, task scheduling, and resource allocation for task offloading with edge-cloud cooperation," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 238–256, Jan. 2024.

[37] W. Zhao et al., "DRL connects Lyapunov in delay and stability optimization for offloading proactive sensing tasks of RSUs," *IEEE Trans. Mobile Comput.*, vol. 23, no. 7, pp. 7969–7982, Jul. 2024.

[38] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Service placement and request routing in MEC networks with storage, computation, and communication constraints," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1047–1060, Jun. 2020.

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

[40] Z. Liang, Y. Liu, T.-M. Lok, and K. Huang, "Multi-cell mobile edge computing: Joint service migration and resource allocation," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 5898–5912, Sep. 2021.

[41] B. Gao, Z. Zhou, F. Liu, F. Xu, and B. Li, "An online framework for joint network selection and service placement in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 11, pp. 3836–3851, Nov. 2022.

[42] C. Zhang and H. Du, "DMORA: Decentralized multi-SP online resource allocation scheme for mobile edge computing," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2497–2507, Oct.–Dec. 2022.

[43] N. Zhang, Y. Liu, X. Mu, W. Wang, and A. Huang, "Queue-aware STAR-RIS assisted NOMA communication systems," *IEEE Trans. Wireless Commun.*, vol. 23, no. 5, pp. 4786–4801, May 2024.

[44] M. J. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. Williston, VT, USA: Morgan and Claypool Publishers, 2010.

[45] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[46] S. J. Wright, "Coordinate descent algorithms," *Math. Program.*, vol. 151, no. 1, pp. 3–34, 2015.

[47] D. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006.

[48] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[49] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Foundations Trends Netw.*, vol. 1, no. 1, pp. 1–144, 2006.

[50] S. Shen et al., "SAC-PP: Jointly optimizing privacy protection and computation offloading for mobile edge computing," *IEEE Trans. Netw. Service Manag.*, vol. 21, no. 6, pp. 6190–6203, Dec. 2024.