Cost-Effective Traffic Scheduling and Resource Allocation for Edge Service Provisioning

Zhengzhe Xiang, *Member, IEEE*, Yuhang Zheng, Zengwei Zheng, Shuiguang Deng, *Senior Member, IEEE*, Minyi Guo, *Fellow, IEEE*, and Schahram Dustdar, *Fellow, IEEE*

***Abstract*— The multi-access edge computing (MEC) paradigm has emerged as a critical solution to address the exponential growth in mobile web services and devices. By implementing an edge-based service provisioning system (EPS) with servers located at the network's edge, both transmission and computation efficiency can be significantly enhanced. Nevertheless, it is also essential to carefully consider the resource allocation for services, the traffic management of requests, and the path arrangement for data delivery to ensure the cost-effective operation of the EPS. Therefore, we investigate and quantify the relationship between the performance and cost of the EPS in this paper, and model the cost-effective service provisioning problem as a multi-phase convex optimization problem. An online algorithm whose name is `RDC` based on the Lyapunov framework is proposed to decompose this problem into several sub-problems.Additionally, a heuristic approach that partitions edge servers into several clusters, called `RDC-NeP` and based on `RDC`, has also been proposed to reduce computational complexity. A series of experiments were conducted to evaluate the proposed approach. The results demonstrate that `RDC` can effectively balance expense and performance, while `RDC-NeP` significantly simplifies the processing of `RDC` when the problem scale increases.**

***Index Terms*— Multi-access edge computing, resource allocation, service scheduling, service computing.**

I. Introduction

THE recent proliferation of mobile services and devices has necessitated the evolution of traditional mobile computing technologies. According to a report[1] released by HUAWEI Technologies in March 2022, the total number of global connections is projected to exceed 200 billion by 2030. Furthermore, a report[2] by the Global System for Mobile Communications Association predicts that the number of worldwide mobile service subscribers will increase at an average annual growth rate of 1.9% until 2025. Additionally, the global cellular IoT connection number is expected to reach 3.2 billion by 2024. Nevertheless, the instability of wireless channels and the limited resources of mobile devices present challenges for users in achieving an efficient and seamless experience. To address these concerns, the multi-access edge computing paradigm (MEC), also known as "mobile edge computing," has been proposed [1], [2], [3]. As an extension of mobile cloud computing, MEC optimizes the resource usage from the near-user side to provide context-aware services [4], [5]. With its help, computation and transmission between mobile devices and the cloud are allowed to be migrated to edge servers.

In the MEC paradigm, users can connect to nearby edge servers via a wireless network to access powerful resources for task completion. This is in contrast to using cloud resources located far away from the users, which can result in higher latency. In addition to reducing latency by localizing computation, edge servers can also work together to make good use of the available resources. Mature Platform-as-a-Service (PaaS) management frameworks, such as EdgeSite,[3] can be used to organize edge servers into a cluster to handle user requests from any others. Moreover, resource allocation and release for services can be easily managed. However, it is important to note that MEC is not a panacea for optimizing mobile computing; while it offers improvements, it relies on external resources that may incur significant costs. Therefore, it is essential for any system design to take into account its total cost and design cost-effective service provisioning strategies [6], [7] in the MEC-based systems to make full use of the limited edge resources, as the cost-performance ratio is a significant factor in determining its viability [8], [9].

In this paper, we address the challenge of designing an effective service provisioning strategy. To this end, we propose a composite strategy comprising three components: resource allocation, traffic scheduling, and data delivery. The paper's main contributions are:

1) The cost-effective service provisioning problem in the MEC environment is formulated as a joint optimization problem that aims to minimize the expected service response time under the constraints of edge server available resource limitation and expense.

2) We introduce an online approach, referred to as `RDC`, to tackle the specified issue. This approach employs a Lyapunov optimization framework and decomposes the target

Manuscript received 2 June 2022; revised 27 December 2022 and 29 March 2023; accepted 1 April 2023; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor C. Wu. Date of publication 24 April 2023; date of current version 19 December 2023. This work was supported by the National Natural Science Foundation of China under Grant 62102350, Grant U20A20173, Grant 62125206, and Grant 62072402. *(Corresponding authors: Zengwei Zheng; Shuiguang Deng.)*

Zhengzhe Xiang is with the School of Computer and Computing Science, Hangzhou City University, Hangzhou 310015, China, and also with the Emerging Parallel Computing Center, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: xiangzz@zucc.edu.cn).

Yuhang Zheng and Shuiguang Deng are with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China (e-mail: zyhxxds_ludwig@zju.edu.cn; dengsg@zju.edu.cn).

Zengwei Zheng is with the School of Computer and Computing Science, Hangzhou City University, Hangzhou 310015, China (e-mail: zhengzw@zucc.edu.cn).

Minyi Guo is with the Emerging Parallel Computing Center, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: guo-my@cs.sjtu.edu.cn).

Schahram Dustdar is with the Distributed Systems Group, TU Wien, 1040 Vienna, Austria (e-mail: dustdar@dsg.tuwien.ac.at).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2023.3265002, provided by the authors.

Digital Object Identifier 10.1109/TNET.2023.3265002

[1]https://www.huawei.com/en/annual-report/2021

[2]https://www.gsmaintelligence.com

[3]https://docs.kubeedge.io/en/latest/modules/edgesite.html

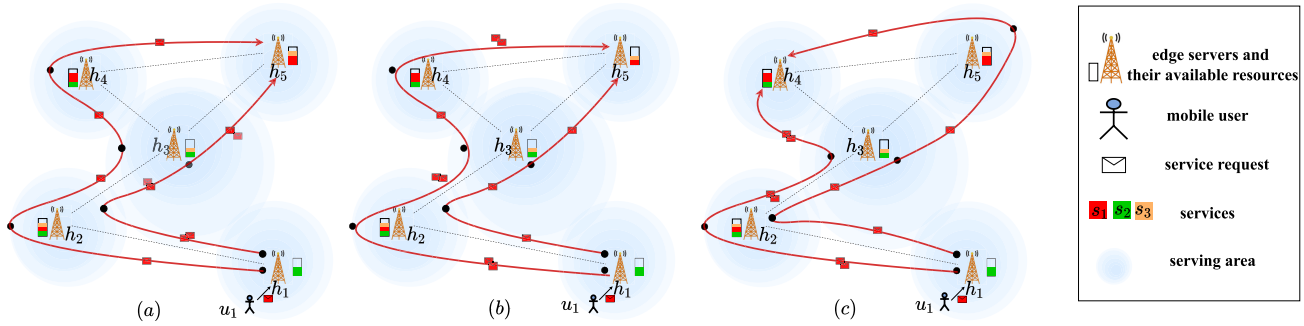

Fig. 1. Examples of a 4-server/3-service/1-user EPS.

problem into three distinct sub-problems. We prove the convexity of these sub-problems, enabling them to be solved separately with ease.

3) In light of the large-scale cases, a heuristic approach known as RDC−NeP is utilized to segregate the nodes into distinct cliques to enable the parallel reuse of RDC.

4) The advantages of our approach are underpinned by a series of comparison experiments with other existing baselines and the advantages are clearly illustrated. What's more, to explore the factors that may affect the RDC's results, we apply the RDC approach on sets of different system configurations and analyze their individual contributions.

The remainder of this article is structured as follows: In Section II, we use a concise yet illustrative example to present the issues that will be addressed in this study. Section III reviews previous research relevant to the identified problems. In Section IV, we introduce the definitions and concepts that form the basis of the problem being proposed. Section V outlines the proposed solutions to address the problem at hand. In Section VI, we explore the challenges that arise when dealing with a large number of edge servers, and present a heuristic solution known as RDC−NeP. Section VII presents the experimental results, including an analysis of the factors that influence our algorithms. Finally, in Section VIII, we conclude our contributions and highlight possible avenues for future research based on the findings.

II. Motivation Scenario

In the context of the edge-based service provisioning system (EPS), the choice of resource allocation and traffic scheduling strategies can significantly impact the performance and cost of the system [10]. This is particularly pertinent when edge server resources are constrained and their cooperation is complex. In this section, we illustrate the importance of these strategies through a succinct yet illustrative example.

As shown in Fig.1, we establish a simple EPS which has five edge servers (h_1, h_2, h_3, h_4, h_5) as a cluster and three types of services (s_1, s_2, s_3) running on the edge servers and providing different functionalities for users in the serving areas of this system. Suppose the user u_1 is going to use service s_1 to process the uploaded data (whose data volume is d) collected by his or her wearable mobile devices, this user will first connect to h_1 (in this work, the nearest-first principle is adopted when there are overlaps between different serving areas). With different colors identifying the service types, different areas of the colorful blocks revealing the amount of their allocated resource, and dash lines among edge servers describing the cluster topology, it can be found that the service instances of s_1 on h_2, h_4, or h_5 can be used to accomplish the task (this is because only these three edge servers have allocated resources to s_1 and they are also reachable from the connected edge server h_1 of user u_1 in the cluster). Now, if u_1 uses the service instance of s_1 on h_5 as what is shown in Fig.1(a) and Fig.1(b), there will be two possible data delivery paths $\phi_1 = h_1 \rightarrow h_2 \rightarrow h_4 \rightarrow h_5$ and $\phi_2 = h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow h_5$ to send corresponding data to that service instance according to the network topology shown with the dash lines. Apparently, the uploading of used data d can be accelerated with parallel transmission: the data d can be separated into two parts d_1 and d_2 ($d_1 + d_2 = d$) and transmitted via these two paths so that the final time that s_1 on h_5 receives the whole d will be estimated with

$$\mathtt{T}_{1,5}(d, \{\phi_1, \phi_2\}) = \max\Big[\mathtt{trans\_time}(d_1, \phi_1), \mathtt{trans\_time}(d_2, \phi_2)\Big] \quad (1)$$

where $\mathtt{trans\_time}(d_i, \phi_i)$ is the transmission time of data packets whose volume is d_i via delivery path ϕ_i. It will be easy to conclude that a lame-duck-like data delivery path selection will result in slow data transmission. Meanwhile, as we use the number of envelopes to describe the data volume in Fig.1, we can also find that the separations of d, namely $d_1 < d_2$ in Fig.1(a) and $d_1 > d_2$ in Fig.1(b), will affect the value of $\mathtt{T}_{1,5}(d, \{\phi_1, \phi_2\})$ as well. Thus, *we need to take care of the data delivery in the EPS* — this is the first important issue we would like to highlight and solve.

After the data is ready for s_1 on h_5, the service instance will start to process it. Given the resource allocated to s_1 by h_5 denoted with $\mathtt{res}_{5,1}$, the processing time $\mathtt{exec\_time}(\mathtt{res}_{5,1})$ will be added to $\mathtt{T}_{1,5}(d, \{\phi_1, \phi_2\})$ to obtain the total time cost of case Fig.1(a) and Fig.1(b):

$$\mathtt{T}^{(a)/(b)} = \mathtt{T}_{1,5}(d, \{\phi_1, \phi_2\}) + \mathtt{exec\_time}(\mathtt{res}_{5,1}) \quad (2)$$

At the same time, as the total cost of example (a) or (b) in Fig.1 strongly depends on the allocated resource $\mathtt{res}_{5,1}$, we can also use $\mathtt{C}^{(a)/(b)}(\mathtt{res}_{5,1})$ to represent it. Therefore, it will be clear that the second key point can be summarized when comparing the performance and cost in Fig.1(a) and Fig.1(b) — the EPS in Fig.1(b) has less resource for s_1 on h_5 (represented by the area of colored blocks), it means that the request processing performance (quantified by the total time cost) is worse than that of Fig.1(a); but just for the less resources the EPS in Fig.1(b) has used, the total cost for the used resource may be less because the cost $\mathtt{C}^{(a)/(b)}(\mathtt{res}_{5,1})$ is usually a monotone and non-decreasing function on the used resource $\mathtt{res}_{5,1}$. Namely, *there will be a trade-off between the performance and cost of a given EPS.*

At last, for the case where u_1 choose the service s_1 on h_4, instead of h_5, to process the data as shown in Fig.1(c), though h_4 can allocate the same resource to s_1 on it as that of h_5, but the possible data delivery paths changes to $\phi_3 = h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow h_4$ and $\phi_4 = h_1 \rightarrow h_2 \rightarrow h_3 \rightarrow h_5 \rightarrow h_4$, which results in the changing of transmission time estimation. Thus, *the traffic scheduling of service requests will also matter in optimizing the system performance*.

III. Related Work

In this section, we will review some representative works that are related to our problem to see what efforts are made to optimize the performance and cost of service provisioning systems in the MEC environment.

A. Resource Allocation in Service Provisioning

Resource allocation is a long-standing and significant concern that is extensively discussed in research on distributed systems to achieve optimal performance and cost balance.

For example, in the environments where the MEC paradigm is used for performance improvement, the researchers care more about the effective running of the services: Brik et al. [11] introduced an algorithm to address the challenge of balancing computational load among mobile edge platforms by leveraging the Tabu-Search meta-heuristic to solve the service resource allocation problem, subject to constraints on computing resources and latency. Subsequently, Moubayed et al. [12] and Liu et al. [13] extended this work to future intelligent transportation systems, where they formulated the problem of optimal vehicle service deployment in a hybrid core/edge/vehicle environment as an optimization problem. This optimization problem determines whether to allocate resources to services or not, and they developed approximate algorithms to solve this problem. On the other hand, in environments where the issue of energy saving is on the table, the researcher will spare no effort to consider completing tasks with minimum energy consumption. Bi et al. [14] further explored a similar topic, they devised a mixed integer non-linear programming that jointly optimized service caching, computation offloading decisions, and system resource allocation. They employed a derived closed-form expression for optimal resource allocation to convert the problem into an equivalent pure binary integer linear programming. Additionally, the researchers utilized the underlying structures of caching causality and task dependency models, which enabled them to develop a reduced-complexity alternating minimization technique. This technique facilitated the updating of caching placement and offloading decisions alternately, ultimately reducing time cost and energy consumption. In order to achieve service resource allocation in a highly dynamic mobile network, Ning et al. [15] proposed a sample average approximation-based stochastic algorithm. This algorithm approximates the future expected system utility, which is the sum of energy consumption of different periods. They then utilized a distributed Markov approximation algorithm to determine the service deployment configurations. Considering the fact that too many small cells may increase operational costs and emit more CO_2, Mohajer et al. [16] proposes a dynamic optimization model which maximizes the total uplink/downlink energy efficiency along with satisfying the necessary QoS constraints by associating user equipment with the right cells.

B. Traffic Scheduling in Service Provisioning

Traffic scheduling issues arise due to the distribution and redundancy of resources, necessitating judicious decision-making when faced with alternatives. Optimal choices depend on the objectives under consideration, leading to the study of how to satisfy demands from various perspectives.

Some researchers analyze the structure of their target system and propose different traffic scheduling approaches to ensure the flexibility of network services at the network edge: Poularakis et al. [17] considered the situations that emerging services exhibit asymmetric bandwidth requirements. Besides the joint optimization of service resource allocation, they further studied request routing in dense MEC networks with multidimensional constraints and proposed an algorithm that achieved close-to-optimal performance using a randomized rounding technique. Liu et al. [18] studied the joint service function chain deployment and resource management problem in heterogeneous edge environments to minimize the total system delay. Based on the game theory, they proposed an intelligent approach to deploy service function chains and manage resources. Akhtar et al. [19] investigated the problem of optimizing the layout of application function pipes and directing traffic through them on a multi-technology edge network model consisting of wired and wireless millimeter wave links. They use a comprehensive "microscopic" binary integer program to model the system, and a heuristic to achieve high data rates and low latency. Another related study from Kim et al. [20] focuses on queuing delay of mobile devices, it investigated new self-learning network management algorithms at edge nodes, proposed an approach based on deep reinforcement learning techniques, and introduced a scaling factor in the reward function to achieve a trade-off between queuing delay and throughput.

Some other researchers try to optimize the basic facilities by introducing novel traffic scheduling strategies: Bukhsh et al. [21] have used decentralized techniques to process tasks in a parallel manner and build groups of edge nodes when they find cluster management and grouping lead to additional overhead. In distributed edge computing networks, local information of edge nodes is used by them to improve the reliability of the network while a high-availability technique is proposed to enhance the overall edge computing environment. Also, they propose a latency-aware algorithm for edge computing with high availability to detect edge node failures, repair edge nodes, and replace edge nodes with backups. Aiming at the problem of edge system fault handling, Ergun et al. [22] proposed new dynamic reliability management technology for edge computing systems to meet the quality of service requirements while maximizing the remaining energy of the devices.

These studies elucidate fundamental concepts and inspire optimization approaches for service provisioning systems in MEC environments. Building on this work, we aim to balance performance and cost in MEC service provisioning systems by developing a joint allocation, delivery, and scheduling strategy that optimizes system performance while ensuring reasonable expenses.

IV. System Model and Problem Description

In summary, the EPS serves as the governing body for edge servers and their corresponding services, which are responsible for managing user requests at the network's edge. In this

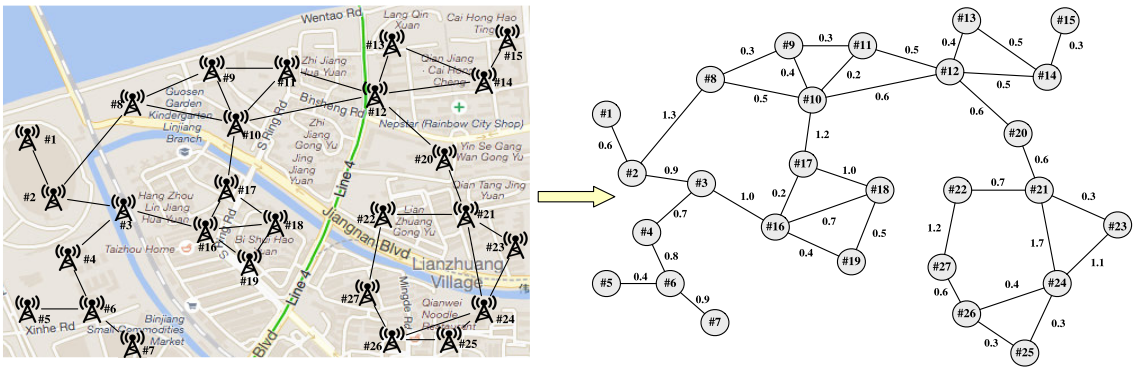

Fig. 2. Architecture of mobile edge computing (left) Using an undirected graph to describe the structure of the network (right).

section, we shall explicate our approach to modeling the entities and their respective behaviors within the EPS, followed by a formulation of the target problem based on these introduced concepts. To make it easier to follow the formulation, we've provided a notation table for quick reference in the notation table provided in supplementary material.

In this paper, we endeavor to employ a symbol system to precisely depict the problem at hand. Accordingly, we hereby establish a uniform convention, whereby the variable i denotes the index of services, j and k denote the indices of servers, p denotes the index of data paths, and q denotes the index of servers within a data path. On the other hand, to differentiate between the variables that may change in the data forwarding (uploading data for processing) and backing (getting the computation results), we'll employ the symbols "$\rightarrow$" and "$\leftarrow$" to mark them accordingly. For example, when seeing a variable $\mu_{k,i}$ you will catch that it is related to the i-th service and the k-th edge server, and will keep still in the forwarding and backing phases. As another illustration, whenever encountering the variable $\overleftarrow{\varrho}_{i,k,j,p}$, it should be noted that it pertains to the i-th service, the k-th and j-th edge server, and the p-th path, and furthermore, it is exclusively employed in the "backing" phase.

A. EPS Entity Modelling

Suppose $\mathcal{H} = \{h_1, h_2, \ldots, h_N\}$ is the set of the widely distributed edge servers. For the users ($\mathcal{U}$) in the serving area of edge servers in $\mathcal{H}$, they will send service requests about services in $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$ to their nearby edge servers to fulfill different tasks. Here $\mathcal{S}$ is the set of services that a service provider has. In a typical MEC environment, if corresponding service instances are running on these edge servers, the nearby edge servers will handle these requests. What's more, as mentioned above, these edge servers may also cooperate with each other to make full use of the edge resources — they can dispatch the service requests to more appropriate edge servers via the network links among them. If we denote the network link between the j-th edge server (h_j) and the k-th edge server (h_k) with $e_{j,k}$, then we can describe those edge servers with a graph $\mathcal{G} = (\mathcal{H}, \mathcal{E})$ like Fig.2, where $\mathcal{E} = \{e_{j,k} \mid h_j, h_k \in \mathcal{H}\}$. Therefore, an EPS can be represented with a 3-tuple EPS = $(\mathcal{G}, \mathcal{S}, \mathcal{U})$.

B. EPS Performance Evaluation

To assess the performance of an EPS, we calculate the average latency of any given service request, which encompasses five constituent parts:

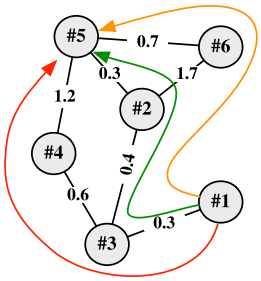

Fig. 3. An example to show the process of data delivery. The access server is h_1 and the executor server is h_5 in this case, and there are 3 possible delivery paths: $\phi_1^{1,5} = (h_1, h_3, h_4, h_5)$, $\phi_2^{1,5} = (h_1, h_3, h_2, h_5)$, $\phi_1^{1,5} = (h_1, h_3, h_2, h_6, h_5)$. When we set $\overrightarrow{\varrho}_{i,1,5,1} = 0.2$, $\overrightarrow{\varrho}_{i,1,5,2} = 0.3$, $\overrightarrow{\varrho}_{i,1,5,3} = 0.5$, it means 20%, 30% and 50% of the related data in s_i's request will go through $\phi_1^{1,5}$, $\phi_2^{1,5}$ and $\phi_3^{1,5}$ individually from h_1 to h_5. The executor server will use a service instance of s_i on it to fulfill the corresponding task when data of these 3 paths are all received by it.

1) Edge Access, EA: First, when a user u intends to execute a particular task associated with service s_i, he/she will connect and send a related service request to the nearby edge server h_j (namely, the *access server*). This radio access network (RAN) latency can be estimated with:

$$\ell_{i,j}^{A} = \frac{I_i}{\nu_j} \tag{3}$$

if the average data size of s_i's input is denoted with I_i and ν_j is the wireless transmission rate between edge server h_j and its users.

2) Service Scheduling, SS: Then, h_j will dispatch the request of s_i (with the input data) to h_k (we can call it *executor server*) via possible delivery paths $\phi_p^{j,k} \in \Phi^{j,k}$ in parallel under the delivery strategy $\overrightarrow{\boldsymbol{\varrho}}$. Here the sequence $\phi_p^{j,k} = (h_j, \hbar_1^{j,k,p}, \hbar_2^{j,k,p}, \ldots, \hbar_{n_{j,k,p}}^{j,k,p}, h_k)$ is used to denote the p-th possible delivery path from h_j to h_k in graph $\mathcal{G}$, and $n_{j,k,p}$ is the number of intermediate edge servers the path $\phi_p^{j,k}$ has. For all $q \in [1, n_{j,k,p}]$, the edge server $\hbar_q^{j,k,p} \in \mathcal{H}$ acts as an intermedia to deliver the request (it is obvious that $\forall \hbar_x^{j,k,p}, \hbar_y^{j,k,p} \in \phi_p^{j,k}$ and $x \neq y$, we will have $\hbar_x^{j,k,p} \neq \hbar_y^{j,k,p}$ to ensure that there is no loop in all delivery paths). An example is shown in Fig.3.

Therefore, by denoting data delivery strategy $\overrightarrow{\varrho}_{i,j,k,p}$ as how many percentages of service s_i's input data is transmitted through $\phi_p^{j,k}$ in this phase constrained by:

$$\begin{aligned} &\sum_{p=1}^{|\Phi^{j,k}|} \overrightarrow{\varrho}_{i,j,k,p} = 1, \forall i, j, k \\ &0 \leq \overrightarrow{\varrho}_{i,j,k,q} \leq 1, \forall i, j, k, p \end{aligned} \tag{4}$$

Therefore, this delivery latency can be represented with:

$$\ell^R_{i,j,k} = \max_{\phi_p^{j,k} \in \Phi^{j,k}} \{\ell^P_{j,k,p} + \ell^T_{i,j,k,p}\}, \tag{5}$$

where $\ell^P_{j,k,p}$ is the propagation latency of the network, and $\ell^T_{i,j,k,p}$ is the transmission latency for delivering s_i's input data through $\phi_p^{j,k}$. As is known, the propagation latency is strongly related to the distance between both ends. For example, there will be about 5.0 μs of propagation latency for every kilometer.[4] In practice, things will be more complex, because the optical fiber between edge servers is rarely straight lines. But we can still use an equivalent parameter κ to depict the relationship between the actual propagation latency and the distance — suppose $\mathscr{D}\langle h_x, h_y\rangle$ is the distance between h_x and h_y, the propagation latency of $e_{x,y}$ can be represented with the following constraints:

$$latency_{x,y} = \kappa \cdot \mathscr{D}\langle h_x, h_y\rangle \tag{6}$$

Then, denote h_j and h_k with $\hbar_0^{j,k,p}$ and $\hbar_{n_{j,k,p}+1}^{j,k,p}$ in $\phi_p^{j,k}$, the propagation latency $\ell^P_{j,k,p}$ can be represented with:

$$\ell^P_{j,k,p} = \kappa \cdot \sum_{q=0}^{n_{j,k,p}} \mathscr{D}\langle \hbar_q^{j,k,p}, \hbar_{q+1}^{j,k,p}\rangle \tag{7}$$

Besides this, suppose $\mathscr{B}\langle h_x, h_y\rangle$ (in mbps) is the bandwidth of link $e_{x,y}$, the transmission latency $\ell^T_{i,j,k,p}$ is:

$$\ell^T_{i,j,k,p} = \sum_{q=0}^{n_{j,k,p}} \frac{I_i \cdot \overrightarrow{\varrho}_{i,j,k,p}}{\mathscr{B}\langle \hbar_q^{j,k,p}, \hbar_{q+1}^{j,k,p}\rangle} \tag{8}$$

3) Service Execution, SE: Now, the executor server h_k will process the service request about s_i. Suppose w_i is the average workload for service s_i (in Million Instructions, MI), the actual service request arrival rate about s_i on h_k is $\lambda_{k,i}$, and the processing capacity for s_i on h_k is $\mu_{k,i}$ (in Million Instructions per Second, MIPS), it can handle $\gamma_{k,i} = \mu_{k,i}/w_i$ requests for s_i per second in average, then the service latency can be represented with Little's Law if the service requests are described with the M/M/1 model [23]:

$$\ell^E_{i,k} = \frac{1}{\gamma_{k,i} - \lambda_{k,i}} \tag{9}$$

It is worth noting that we should keep

$$\frac{\lambda_{k,i}}{\gamma_{k,i}} < 1 \tag{10}$$

in the serving queue to avoid congestion. Here, the M/M/1 model is used because it is a common model which has been widely used in describing service systems like the works of [24], [25], and [26]. To validate the rationality of this model, here we also analyze the UMass network trace dataset[5] which contains a collection of user requests for specific services [27]. The histogram of the time intervals of the neighboring service requests is shown in Fig.4.

We can find in this figure that the assumption of the service request as a Poisson stream is very close to the real world, so the selected model will be able to describe their behaviors with it. But it is also necessary to be informed that the M/M/1 is not the only suitable model, the followers can replace it with other queue models like M/G/1 and M/E$_k$/1 to extend the capability of the approach. Besides this, if the maximum resource of h_k is $\mu_k^\star$, then $\forall k \in [1, N]$, we have:

$$0 \le \mu_{k,i} \tag{11}$$

$$\sum_{i=1}^{M} \mu_{k,i} \le \mu_k^\star \tag{12}$$

the reason we can represent the constraints of resource limitation like this is that the sum of K Poisson distributions whose parameters are λ_1-λ_K is still a Poisson distribution, and the parameter of it is $\sum_i^K \lambda_i$.

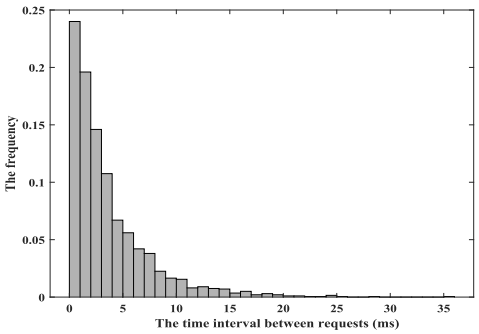

Fig. 4. The histogram of time intervals between requests.

4) Edge Back, EB: After processing the service request on the executor server h_k, the expected result of s_i whose average data size is O_i will be obtained and it needs to be sent back to the user via the access server h_j. Similarly, by denoting data delivery strategy $\overleftarrow{\varrho}_{i,k,j,p}$ as how many percentage of service s_i's input data is transmitted through $\phi_p^{k,j}$ in this phase, the routing latency $\ell^B_{k,j}$ will be:

$$\ell^B_{i,k,j} = \max_{\phi_p^{k,j} \in \Phi^{k,j}} \{\ell^P_{k,j,p} + \ell^T_{i,k,j,p}\}, \tag{13}$$

where corresponding propagation and transmission latency can be represented with:

$$\ell^P_{k,j,p} = \kappa \cdot \sum_{q=0}^{n_{k,j,p}} \mathscr{D}\langle \hbar_q^{k,j,p}, \hbar_{q+1}^{k,j,p}\rangle \tag{14}$$

$$\ell^T_{i,k,j,p} = \sum_{q=0}^{n_{k,j,p}} \frac{O_i \cdot \overleftarrow{\varrho}_{i,k,j,p}}{\mathscr{B}\langle \hbar_q^{k,j,p}, \hbar_{q+1}^{k,j,p}\rangle} \tag{15}$$

5) Callback, CB: Finally, the access server, which is connected to the user u, will send the received result of s_i to that user's device. Obviously, this transmission latency is:

$$\ell^U_{i,j} = \frac{O_i}{\nu_j} \tag{16}$$

Now, the total service latency for a service s_i's request, which is received by h_j and processed by h_k, will be:

$$\ell_{i,j,k} = \quad \ell^A_{i,j} + \ell^R_{i,j,k} + \ell^E_{i,k} + \ell^B_{i,j,k} + \ell^U_{i,j} \tag{17}$$

[4] https://en.wikipedia.org/wiki/Latency_(engineering)
[5] http://skulddata.cs.umass.edu/traces/network

C. The Pricing Model of EPS

In the end, service providers must pay for the used resources, but the final expense depends on the actual pricing models of the infrastructure vendors [10]. These models determine the relationship between price and resource usage. Vendors typically employ diverse pricing models based on their marketing strategies to enhance competitiveness. Regardless of how these models may change, the underlying principle remains constant: greater resource usage incurs higher costs, reflecting an on-demand pricing model. In our work, we employ a representative linear pricing model to evaluate expenses for processed workloads using allocated computational resources:

$$C(\boldsymbol{\mu}) = \sum_{k=1}^{N}\sum_{i=1}^{M} \eta_k \mu_{k,i}. \tag{18}$$

when η_k is denoted as the unit price for using resources of h_j. As our work focuses solely on computational resources, readers can readily extend the on-demand pricing model by incorporating additional resource costs.

D. Problem Formulation

As described above, we have a complete and detailed analysis of the latency for an arbitrary service request. To go a step further, if we use $\vartheta_{i,j,k}$ to denote the routing strategy which describes the probability of "h_j choose h_k for processing when receiving requests of s_i", we can also get the probability of $< i,j,k >$, the event "h_j receives the request about s_i and dispatches it to h_k for processing":

$$\begin{aligned} Pr(<i,j,k>) &= Pr(h_k|s_i,h_j)\cdot Pr(s_i|h_j)\cdot Pr(h_j) \\ &= \vartheta_{i,j,k}\cdot \frac{\sum_{z=1}^{M}\Lambda_{z,j}}{\sum_{i=1}^{M}\sum_{j=1}^{N}\Lambda_{i,j}}\cdot\frac{\Lambda_{i,j}}{\sum_{z=1}^{M}\Lambda_{z,j}} \\ &= \vartheta_{i,j,k}\cdot\frac{\Lambda_{i,j}}{\Lambda_U}, \end{aligned} \tag{19}$$

where $\Lambda_{i,j}$ means the average service request arrival rate for service s_i (in req/s) on h_j, and $\Lambda_U = \sum_{i=1}^{M}\sum_{j=1}^{N}\Lambda_{i,j}$ is the total service request arrival rate of all the users served by $\mathcal{H}$. It is obvious that we will have the following constraints for $\boldsymbol{\vartheta}$, $\forall(i,j)\in[1,M]\times[1,N]$:

$$\sum_{k=1}^{N}\vartheta_{i,j,k} = 1 \tag{20}$$

$$0 \le \vartheta_{i,j,k} \le 1 \tag{21}$$

while $\lambda_{k,i}$, the actual request arrival rate about s_i on host h_k that can be inferred recursively with $\{\Lambda_{*,*}\}$ according to the Burke's theorem [28]:

$$\lambda_{k,i} = \sum_{z=1}^{N}\vartheta_{i,z,k}\Lambda_{z,i}. \tag{22}$$

Therefore, for a given edge service provisioning system EPS = $(\mathcal{G},\mathcal{S},\mathcal{U})$, the expectation of service latency can be represented with Equation (23) and expanded in (24), as shown at the bottom of the page

$$\mathbb{E}_\ell(\boldsymbol{\vartheta},\boldsymbol{\varrho},\boldsymbol{\mu}) = \sum_{i,j,k} Pr(s_i,h_j,h_k)\cdot\ell_{i,j,k} \tag{23}$$

According to the expressions of $Pr(s_i,h_j,h_k)$ shown in Equation (19) and that of $\ell_{i,j,k}$ shown in Equation (3) - (17), the value of it can be further measured with Equation (24), where in Equation (23) and (24) $\boldsymbol{\mu}$ is the matrix containing all the $\{\mu_{k,i}\}$ for $\forall(k,i)\in[1,N]\times[1,M]$ and $\boldsymbol{\varrho}$ denotes the combination of $\overrightarrow{\boldsymbol{\varrho}}$ and $\overleftarrow{\boldsymbol{\varrho}}$ who have $\{\overrightarrow{\varrho}_{i,j,k,p}\}$ and $\{\overleftarrow{\varrho}_{i,k,j,p}\}$ as their elements separately.

Due to the dynamic nature of users' service requirements, static models are insufficient for characterizing and assessing the performance costs of systems over time. Hence, inspired by the works like [29] and [30], we separate time into several discrete time slots to evaluate the EPS over a long time period, where each time slot has an appropriate duration that matches the timescale in which the service provisioning strategy can be updated. In this way, if we rewrite the above $\mathbb{E}_\ell(\boldsymbol{\vartheta},\boldsymbol{\varrho},\boldsymbol{\mu})$ with $\mathbb{E}_\ell^t(\boldsymbol{\vartheta}^t,\boldsymbol{\varrho}^t,\boldsymbol{\mu}^t)$ to represent the average service latency in the t-th time slot, and rewrite $C(\boldsymbol{\mu})$ as $C^t(\boldsymbol{\mu}^t)$, $\Lambda_{i,j}$ as $\Lambda_{i,j}^t$ etc. in the same way, then the average latency of a service request can be described as follows:

$$\hat{\ell}(\boldsymbol{\vartheta},\boldsymbol{\varrho},\boldsymbol{\mu}) = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}_\ell^t(\boldsymbol{\vartheta}^t,\boldsymbol{\varrho}^t,\boldsymbol{\mu}^t) \tag{25}$$

and the budget constraint can be described with:

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}C^t(\boldsymbol{\mu}^t) \le C^\star \tag{26}$$

In summary, the EPS performance optimization problem can be described as follows:

$$\begin{aligned} \mathcal{P}_1: \quad & \min_{\boldsymbol{\vartheta},\boldsymbol{\varrho},\boldsymbol{\mu}} \hat{\ell}(\boldsymbol{\vartheta},\boldsymbol{\varrho},\boldsymbol{\mu}) \\ & s.t. \quad (4),(10),(12),(20),(21),(26) \end{aligned} \tag{27}$$

It is worth noting that there in Equation (25) and Equation (26), $\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}g^t(x)$ is used to represent the long-term time-average of function $g(x)$ of a time sequence to describe the performance and expense. Furthermore, to avoid a large variance of the sequence $\{g^t(x)\}_{t=0}^{T-1}$, we use constant $C^\star$ in Equation (26) so that the expense in every time slot is acceptable. This constraint ensures a stable and predictable financial expenditure for the service providers.

V. The RDC Algorithm and Analysis

In this section, we will introduce the approach to the request routing, data delivery, and resource allocation (RDC) problem. The proposed approach or algorithm shares the same

$$\begin{aligned} \mathbb{E}_\ell(\boldsymbol{\vartheta},\boldsymbol{\varrho},\boldsymbol{\mu}) = \frac{1}{\Lambda_U}\sum_{i,j,k}\Bigg(&\max_{\phi_p^{j,k}\in\Phi^{j,k}}\left\{\sum_{q=0}^{n_{j,k,p}}\left[\kappa\cdot\mathscr{D}\langle\hbar_q^{j,k,p},\hbar_{q+1}^{j,k,p}\rangle + \frac{I_i\cdot\overrightarrow{\varrho}_{i,j,k,p}}{\mathscr{B}\langle\hbar_q^{j,k,p},\hbar_{q+1}^{j,k,p}\rangle}\right]\right\} \\ &+\max_{\phi_p^{k,j}\in\Phi^{k,j}}\left\{\sum_{q=0}^{n_{k,j,p}}\left[\kappa\cdot\mathscr{D}\langle\hbar_q^{k,j,p},\hbar_{q+1}^{k,j,p}\rangle + \frac{O_i\cdot\overleftarrow{\varrho}_{i,k,j,p}}{\mathscr{B}\langle\hbar_q^{k,j,p},\hbar_{q+1}^{k,j,p}\rangle}\right]\right\} + \frac{I_i+O_i}{\nu_j} + \frac{1}{\gamma_{k,i}-\lambda_{k,i}}\Bigg)\vartheta_{i,j,k}\Lambda_{i,j} \end{aligned} \tag{24}$$

name RDC with the problem for simplification. Basically, the RDC algorithm is developed under the Lyapunov optimization framework [29], [31], [32]. With the help of this framework, the long-term optimization problem can be translated into an optimization problem for every slot with a controllable optimum loss.

A. Lyapunov-Based Optimization Framework

In order to generate $\boldsymbol{\vartheta}$, $\boldsymbol{\varrho}$ and $\boldsymbol{\mu}$ in every time slot, we should solve the problem $\mathcal{P}_1$. It can be found that the constraint of long-term average expense shown in Constraint (26) is denoted over all the time slots while those constraints from (4), (10), (12), (20), (21) are claimed in every time slot. So we would like to find out whether the long-term constraint can be translated into an every-slot constraint. First, we will create an expense deficit queue c which is virtual and is used to control the expense per slot. Suppose $c(t)$ is the queue backlog ($c(0) = 0$) in the t-th time slot that describes the current deviation of current expense according to the constraint satisfactory, then the backlog of queue c can be estimated iteratively [10] with:

$$c(t+1) = \max[c(t) + C^t(\boldsymbol{\mu}^t) - C^\star, 0] \tag{28}$$

It is easy to be found that the Constraint (26) will hold when the queue c is *mean rate stable* ($\lim_{t\to\infty} \frac{\mathbb{E}[|c(t)|]}{t} = 0$) so that the problem of managing Constraint (26) will be transformed into a stability control problem for queue c. Therefore, it's necessary to check the conditions that make the expense deficit queue c mean rate stable. We can denote our Lyapunov function which describes c's congestion with

$$L(c(t)) \triangleq \frac{1}{2}c^2(t) \tag{29}$$

According to the Theorem 4.8 and related proofs in [31], we can greedily minimize the drift-plus-penalty (DPP):

$$\text{DPP} = B + V\mathbb{E}_\ell^t(\boldsymbol{\vartheta}^t, \boldsymbol{\varrho}^t, \boldsymbol{\mu}^t) + c(t)(C^t(\boldsymbol{\mu}^t) - C^\star) \tag{30}$$

to generate appropriate $\boldsymbol{\vartheta}^*$, $\boldsymbol{\varrho}^*$ and $\boldsymbol{\mu}^*$ in every time slot, where B is a constant [10], [31] that can be derived from system settings in advance and V is the importance factor. By denoting

$$\begin{aligned}\psi(\boldsymbol{\vartheta}^t, \boldsymbol{\varrho}^t, \boldsymbol{\mu}^t) &= V \cdot \mathbb{E}_\ell^t(\boldsymbol{\vartheta}^t, \boldsymbol{\varrho}^t, \boldsymbol{\mu}^t) + c(t)C^t(\boldsymbol{\mu}^t) \\ &= \frac{V}{\Lambda_U}\sum_{i,j,k}\Big[\overrightarrow{\mathcal{F}}_{i,j,k}(\boldsymbol{\varrho}^t) + \overleftarrow{\mathcal{F}}_{i,j,k}(\boldsymbol{\varrho}^t) + \frac{I_i + O_i}{\nu_j} \\ &\quad + \frac{1}{\gamma_{k,i} - \lambda_{k,i}}\Big]\vartheta_{i,j,k}^t\Lambda_{i,j} + c(t)\sum_{j,i}\eta_j\mu_{j,i}^t \end{aligned} \tag{31}$$

as the objective of the original problem $\mathcal{P}_1$, where the routing latency is denoted as $\overrightarrow{\mathcal{F}}_{i,j,k}(\boldsymbol{\varrho}^t)$ and $\overleftarrow{\mathcal{F}}_{i,k,j}(\boldsymbol{\varrho}^t)$ for simplification:

$$\begin{aligned}\overrightarrow{\mathcal{F}}_{i,j,k}(\boldsymbol{\varrho}^t) = \max_{\phi_p^{j,k}\in\Phi^{j,k}} \sum_{q=0}^{n_{j,k,p}} \Big[\kappa \cdot \mathscr{D}\langle \hbar_q^{j,k,p}, \hbar_{q+1}^{j,k,p}\rangle \\ + \frac{I_i \cdot \overrightarrow{\varrho}_{i,j,k,p}^t}{\mathscr{B}\langle \hbar_q^{j,k,p}, \hbar_{q+1}^{j,k,p}\rangle}\Big]\end{aligned} \tag{32}$$

$$\begin{aligned}\overleftarrow{\mathcal{F}}_{i,k,j}(\boldsymbol{\varrho}^t) = \max_{\phi_p^{k,j}\in\Phi^{k,j}} \sum_{q=0}^{n_{k,j,p}} \Big[\kappa \cdot \mathscr{D}\langle \hbar_q^{k,j,p}, \hbar_{q+1}^{k,j,p}\rangle \\ + \frac{O_i \cdot \overleftarrow{\varrho}_{i,k,j,p}^t}{\mathscr{B}\langle \hbar_q^{k,j,p}, \hbar_{q+1}^{k,j,p}\rangle}\Big]\end{aligned} \tag{33}$$

the one-slot optimization problem can be represented as:

$$\mathcal{P}2: \quad \min_{\boldsymbol{\vartheta}^t, \boldsymbol{\varrho}^t, \boldsymbol{\mu}^t} \psi(\boldsymbol{\vartheta}^t, \boldsymbol{\varrho}^t, \boldsymbol{\mu}^t) \tag{34}$$

$$s.t.\ (4), (10), (12), (20), (21) \tag{35}$$

It is worth noting that it is proved in [31] if $\mathcal{P}2$ is solved with a α-additive approximation [33], the performance of the obtained $\mathcal{P}1$ objective will be guaranteed by:

$$\hat{\ell}(\boldsymbol{\vartheta}, \boldsymbol{\varrho}, \boldsymbol{\mu}) \le \ell^{opt} + \frac{B+\alpha}{V} \tag{36}$$

B. Problem Analysis

After showing the formulation of the RDC problem, we will now investigate its structure to develop an appropriate algorithm and make a detailed analysis.

1) Delivery Path Exploring: It is worth noting that in the RDC problem, request data can be delivered in different paths in parallel, and the path set between h_j and h_k is denoted with $\Phi^{j,k}$. As $\Phi^{j,k}$ is determined by the network structure, we can then get all the path sets for $(j,k) \in [1,N]\times[1,N]$ in advance. The **d**elivery **p**ath **m**ining (DPM) method shown in Algorithm-1 describes the process of finding all the paths of $\Phi^{j,k}$.

Algorithm 1 Delivery Path Mining (DPM)

Input: $\mathcal{G}$ = the graph in which we're to find the path set
h_j: the start server of the data delivery path
h_k: the end server of the data delivery path
Output: Φ: the set of delivery paths

```
1:  if h_j = h_k then
2:      φ ← (h_k)
3:      return {φ}
4:  else
5:      Φ = {}
6:      for h_n ∈ Adjcent(G, h_j) do
7:          G' ← G − {h_j}
8:          Φ' ← DPM(G', h_n, h_k)
9:          for φ_n ∈ Φ' do
10:             φ_s ← (h_j, φ_n)
11:             Φ.add(φ_s)
12:         end for
13:     end for
14:     return Φ
15: end if
```

The DPM algorithm will stop when the start edge server and the end edge server are the same (Line 1-3). Otherwise, suppose h_n is one of the neighbor edge servers of h_j, then the paths in $\Phi^{j,k}$ can be constructed with $h_j \to \phi_p^{n,k}$, where $\phi_p^{n,k} \in \Phi^{n,k}$ is one possible path between h_n and h_k (Line 5-12). Particularly, to ensure that there will be no loops in paths, the DPM algorithm uses the induced graph $\mathcal{G}'$ in Line 7. Therefore, we use DPM($\mathcal{G}$, h_j, h_k) to enumerate all the possible paths between h_j and h_k. To analyze the complexity of DPM algorithm on $\mathcal{G}$, here we assume that $h_j = h_1$ and $h_k = h_N$ in DPM algorithm because we can always renumber the edge servers by setting h_j as the first one and h_k as the last one. Therefore, we can focus on analyzing the paths between h_1 and h_N. Denote $\mathscr{C}_{DPM}(N)$ as the computation cost for

DPM on a fully connected graph $\mathcal{G}$ to evaluate the worst case. According to the process of DPM algorithm, we can easily find the relation between $\mathscr{C}_{DPM}(N)$ and $\mathscr{C}_{DPM}(N-1)$:

$$\mathscr{C}_{DPM}(N) = N - 1 + (N-2)\cdot\mathscr{C}_{DPM}(N-1) \tag{37}$$

With the proof shown in the supplementary material, we can find the computation complexity of DPM is $O((N-2)!)$. Admittedly, the complexity of the approach may not be optimal. However, this is not a significant concern as the DPM algorithm can be executed offline in advance.

2) Optimal Data Delivery Strategy: From Equation (31) we can clearly find that the objective of $\mathcal{P}_2$ is strongly related to the value of $\min\ \overrightarrow{\mathcal{F}}_{i,j,k}(\boldsymbol{\varrho}^t)$ and $\min\ \overleftarrow{\mathcal{F}}_{i,k,j}(\boldsymbol{\varrho}^t)$. Because their values are independent to $(\boldsymbol{\mu}^t,\ \boldsymbol{\theta}^t)$ according to the expressions in Equation (32) and (33), we can try to optimize them individually. Taking the optimization of $\overrightarrow{\mathcal{F}}_{i,j,k}(\boldsymbol{\varrho}^t)$ as an example: with the definition of ∞-norm of vector $\boldsymbol{x}\in\mathbb{R}^{+n}$:

$$||\boldsymbol{x}||_\infty = \max\{x_1, x_2, \ldots, x_n\} \tag{38}$$

the minimum value of $\overrightarrow{\mathcal{F}}_{i,j,k}(\boldsymbol{\varrho}^t)$ can be transformed into:

$$\min\quad ||\mathtt{Diag}(\boldsymbol{w}_{i,j,k})\cdot\overrightarrow{\boldsymbol{\varrho}}^t_{i,j,k} + \boldsymbol{D}_{j,k}||_\infty \tag{39}$$

under the constraint shown in Equation (4), where $\overrightarrow{\boldsymbol{\varrho}}^t_{i,j,k} = (\overrightarrow{\varrho}^t_{i,j,k,1},\ \overrightarrow{\varrho}^t_{i,j,k,2},\ \ldots,\ \overrightarrow{\varrho}^t_{i,j,k,|\Phi^{j,k}|}\ )^T$, $\mathtt{Diag}(\boldsymbol{w}_{i,j,k})$ is diagonalized matrix that comes from vector

$$\boldsymbol{w}_{i,j,k} = \begin{bmatrix} \sum_{q=1}^{n_{j,k,1}} \frac{I_i}{\mathscr{B}\langle\hbar_q^{j,k,1},\hbar_{q+1}^{j,k,1}\rangle} \\ \sum_{q=1}^{n_{j,k,2}} \frac{I_i}{\mathscr{B}\langle\hbar_q^{j,k,2},\hbar_{q+1}^{j,k,2}\rangle} \\ \vdots \\ \sum_{q=1}^{n_{j,k,|\Phi^{j,k}|}} \frac{I_i}{\mathscr{B}\langle\hbar_q^{j,k,|\Phi^{j,k}|},\hbar_{q+1}^{j,k,|\Phi^{j,k}|}\rangle} \end{bmatrix} \tag{40}$$

and

$$\boldsymbol{D}_{j,k} = \begin{bmatrix} \kappa\sum_{q}^{n_{j,k,1}} \mathscr{D}\langle\hbar_q^{j,k,1},\hbar_{q+1}^{j,k,1}\rangle \\ \kappa\sum_{q}^{n_{j,k,2}} \mathscr{D}\langle\hbar_q^{j,k,2},\hbar_{q+1}^{j,k,2}\rangle \\ \vdots \\ \kappa\sum_{q}^{n_{j,k,|\Phi^{j,k}|}} \mathscr{D}\langle\hbar_q^{j,k,|\Phi^{j,k}|},\hbar_{q+1}^{j,k,|\Phi^{j,k}|}\rangle \end{bmatrix} \tag{41}$$

are used for simplification. Note that $\boldsymbol{D}_{j,k}$ is fixed when j and k are given, and it will not contribute to the searching of optimal $\overrightarrow{\boldsymbol{\varrho}}^t_{i,j,k}$. Therefore, we can temporally ignore it and obtain the simplified problem of $\mathcal{P}_N$:

$$\mathcal{P}_{\overrightarrow{NR}}:\quad \min\quad ||\mathtt{Diag}(\boldsymbol{w}_{i,j,k})\cdot\overrightarrow{\boldsymbol{\varrho}}^t_{i,j,k} + \boldsymbol{D}_{j,k}||_\infty$$
$$s.t.\begin{cases} \sum_{p=1}^{|\Phi^{j,k}|} \overrightarrow{\varrho}^t_{i,j,k,p} = 1 \\ 0 \le \overrightarrow{\varrho}^t_{i,j,k,q} \le 1 \end{cases}$$

With slack variables $\boldsymbol{\xi}\in\mathbb{R}$ and auxiliary vector $\boldsymbol{z}\in\mathbb{R}^{|\Phi^{j,k}|}$ introduced, the problem $\mathcal{P}_{NR}$ can be transformed into the following linear programming problem (LP):

$$\mathcal{P}'_{\overrightarrow{NR}}:\quad \min\quad \xi$$
$$s.t.\begin{cases} \mathbb{1}\cdot\overrightarrow{\boldsymbol{\varrho}}^t_{i,j,k} = 1 \\ \mathtt{Diag}(\boldsymbol{w}_{i,j,k})\cdot\overrightarrow{\boldsymbol{\varrho}}^t_{i,j,k} + \boldsymbol{D}_{j,k} + \boldsymbol{z} = \mathbb{1}\cdot\xi \\ \overrightarrow{\boldsymbol{\varrho}}^t_{i,j,k} \ge \boldsymbol{0} \\ \boldsymbol{z} \ge \boldsymbol{0} \end{cases}$$

As a result, the simplex method or the dual simplex method can be applied here to find the optimal solution of $\mathcal{P}'_{\overrightarrow{NR}}$, and it is clear we can solve the problem $\mathcal{P}'_{\overleftarrow{NR}}$ in the same way if we change the variables from $\overrightarrow{\boldsymbol{\varrho}}^t_{i,j,k}$ to $\overleftarrow{\boldsymbol{\varrho}}^t_{i,j,k}$ and change replace the I_i in $\boldsymbol{w}_{i,j,k}$ with O_i. It is worth noting that the problems are not related to any time-varying variables, thus it is not necessary for us to solve them in every time slot.

3) Optimal Request Routing and Resource Allocation: Denote the optimal value derived from $\mathcal{P}'_{\overrightarrow{NR}}$ and $\mathcal{P}'_{\overleftarrow{NR}}$ with $\overrightarrow{F}_{i,j,k}$ and $\overleftarrow{F}_{i,j,k}$, and let $F_{i,j,k} = \overrightarrow{F}_{i,j,k} + \overleftarrow{F}_{i,j,k}$, we will have the reduced problem $(\mathcal{P}_2^{CT*})$:

$$\mathcal{P}_2^{CT*}:\quad \min_{\boldsymbol{\vartheta}^t,\boldsymbol{\mu}^t} \frac{V}{\Lambda_U}\sum_{i,j,k}\Big[\frac{I_i+O_i}{\nu_j} + \frac{1}{\gamma_{k,i}-\lambda_{k,i}} + F_{i,j,k}\Big]\cdot\vartheta^t_{i,j,k}\Lambda_{i,j} + c(t)\sum_{j,i}\eta_j\mu^t_{j,i}$$
$$s.t.\quad (10),(12),(20),(21)$$

To solve $\mathcal{P}_2^{CT*}$, we will first analyze the properties of the objective and the constraints. Denote $\tau_{i,j,k} = (I_i+O_i)/\nu_j + F_{i,j,k}$ to describe the transmission latency, we can obtain the lower bound of $\mathbb{E}^t_\ell(\boldsymbol{\vartheta}^t,\boldsymbol{\varrho}^t,\boldsymbol{\mu}^t)$, as shown at the bottom of the next page.

Here, the equality holds if-and-only-if,

$$\begin{bmatrix} \vartheta^t_{i,1,1} & \vartheta^t_{i,2,1} & \cdots & \vartheta^t_{i,N,1} \\ \vartheta^t_{i,1,2} & \vartheta^t_{i,2,2} & \cdots & \vartheta^t_{i,N,2} \\ \vdots & \vdots & \ddots & \vdots \\ \vartheta^t_{i,1,N} & \vartheta^t_{i,2,N} & \cdots & \vartheta^t_{i,N,N} \end{bmatrix}\begin{bmatrix} \Lambda^t_{1,i} \\ \Lambda^t_{2,i} \\ \vdots \\ \Lambda^t_{N,i} \end{bmatrix} = \begin{bmatrix} \gamma^t_{1,i}-\kappa^t_i \\ \gamma^t_{2,i}-\kappa^t_i \\ \vdots \\ \gamma^t_{N,i}-\kappa^t_i \end{bmatrix} \tag{42}$$

where the variable κ^t_i is

$$\kappa^t_i = \frac{\sum_{k=1}^N\gamma^t_{k,i} - \sum_{k=1}^N\Lambda^t_{k,i}}{N} > 0 \tag{43}$$

Therefore, we can have the lower bound of the objective:

$$\psi(\boldsymbol{\vartheta}^t,\boldsymbol{\varrho}^t,\boldsymbol{\mu}^t) \ge \psi_1(\boldsymbol{\mu}^t) + \psi_2(\boldsymbol{\vartheta}^t) \tag{44}$$

where $\psi_1(\boldsymbol{\vartheta}^t)$ and $\psi_2(\boldsymbol{\mu}^t)$ are denoted with:

$$\psi_1(\boldsymbol{\mu}^t) = \sum_{i=1}^M \frac{VN\sum_{j=1}^N\Lambda^t_{j,i}}{\Lambda^t_U(\sum_{k=1}^N\gamma^t_{k,i} - \sum_{k=1}^N\Lambda^t_{k,i})} + c(t)\Big[\sum_{j,i}\eta_j\mu^t_{j,i}\Big] \tag{45}$$

$$\psi_2(\boldsymbol{\vartheta}^t) = \frac{V}{\Lambda^t_U}\sum_{i,j,k}\vartheta^t_{i,j,k}\Lambda^t_{j,i}\tau_{i,j,k} \tag{46}$$

and the gap between $\psi(\boldsymbol{\vartheta}^t,\boldsymbol{\varrho}^t,\boldsymbol{\mu}^t)$ and its lower bound $\Delta = \psi(\boldsymbol{\vartheta}^t,\boldsymbol{\varrho}^t,\boldsymbol{\mu}^t) - \psi_1(\boldsymbol{\mu}^t) - \psi_2(\boldsymbol{\vartheta}^t)$ will be at most:

$$\Delta = V\sum_{i=1}^M\left(\sum_{j,k}\frac{\vartheta^t_{i,j,k}\Lambda_{j,i}}{\gamma^t_{k,i}-\lambda_{k,i}} - \frac{N\sum_{k=1}^N\Lambda_{k,i}}{\sum_{k=1}^N\gamma_{k,i} - \sum_{k=1}^N\Lambda_{k,i}}\right)$$
$$\le \sum_{i=1}^M\left(\frac{V\lambda_{1,i}}{\gamma_{1,i}-\lambda_{1,i}} + \ldots + \frac{V\lambda_{N,i}}{\gamma_{N,i}-\lambda_{N,i}} - \frac{NV\sum_{k=1}^N\Lambda_{k,i}}{\sum_{k=1}^N\gamma_{k,i} - \sum_{k=1}^N\Lambda_{k,i}}\right)$$

Based on the stable queue constraint (10), $\forall \epsilon > 0$, we can also control the value of $\gamma_{k,i}/\lambda_{k,i}$ to make it greater than or equal to 1 + ϵ. Thus, we will have:

$$\begin{aligned}\Delta &\leq \frac{VM}{\epsilon} - \sum_{i=1}^{M} \frac{VN^2\Lambda_\circ}{\sum_{k=1}^{N}\gamma_{k,i} - \sum_{k=1}^{N}\Lambda_{k,i}} \\ &\leq \frac{VM}{\epsilon} - \frac{VN^2M^2\Lambda_0}{\sum_{i=1}^{M}\sum_{k=1}^{N}\gamma_{k,i} - \sum_{i=1}^{M}\sum_{k=1}^{N}\Lambda_{k,i}} \\ &\overset{(12)}{\leq} \frac{VM}{\epsilon} - \frac{VN^2M^2\Lambda_\circ}{\hat{\mu}^\star/w_\circ - \Lambda_U} = O(1/\epsilon) \end{aligned} \tag{47}$$

where $\hat{\mu}^\star = \sum_{k=1}^{N}\mu_k^\star$ is the total available resources of the system, $\Lambda_\circ = \min_{j,i}\Lambda_{j,i}$ and $w_\circ = \min_i w_i$ are used here for simplification. At the same time, given an origin problem P_{old}: $\min_{\boldsymbol{x}} f(\boldsymbol{x})$ where $f(\boldsymbol{x}) \geq g(\boldsymbol{x})$ holds for all $\boldsymbol{x} \in \mathbf{Dom}f$, and the gap between these two functions is guaranteed by $f(\boldsymbol{x})$ - $g(\boldsymbol{x}) \leq \Delta$, the difference of the f and g's optimal minimum values for $\boldsymbol{x} \in \mathbf{Dom}f$ is:

$$f(\boldsymbol{x}_f^*) - g(\boldsymbol{x}_g^*) \leq f(\boldsymbol{x}_g^*) - g(\boldsymbol{x}_g^*) \leq \Delta \tag{48}$$

Therefore, we can greedily translate the problem $\mathcal{P}2$ into minimizing the right-hand side lower bound of $\psi(\boldsymbol{\vartheta}^t, \boldsymbol{\varrho}^t, \boldsymbol{\mu}^t)$ in Equation (44) while adding the constraint (43) with an objective value bias of $O(1/\epsilon)$ according to Inequality (47), and the performance gap in Equation (36) can be updated with:

$$\hat{\ell}(\boldsymbol{\vartheta}, \boldsymbol{\varrho}, \boldsymbol{\mu}) \leq \ell^{opt} + \frac{B + O(1/\epsilon)}{V} \tag{49}$$

It is worth noting that in Equation (44) the right-hand side lower bound can be decomposed into two parts that are related to $\boldsymbol{\mu}^t$ and $\boldsymbol{\vartheta}^t$ individually. Therefore, we can finally decompose the problem $\mathcal{P}2$ into 2 subproblems — *optimal resource allocation (ORA)* and *optimal request routing (ORR)*, and the optimal solution can be found when these two problems are solved:

Optimal Resource Allocation (ORA). In this subproblem, the optimal resource $\boldsymbol{\mu}^t$ allocated to services on different hosts can be obtained by solving $\mathcal{P}_2^{ORA}$:

$$\mathcal{P}_2^{ORA}: \quad \min_{\boldsymbol{\mu}^t} \quad \psi_1(\boldsymbol{\mu}^t) \quad s.t. \quad (11), (12), (43) \tag{50}$$

As $\forall \boldsymbol{\mu}_{*,i} \in \mathbf{Dom}\psi_1$, we have:

$$\psi_1(\boldsymbol{\mu}^t) = \sum_{i=1}^{M}\psi_1(\boldsymbol{\mu}_{*,i}^t) \tag{51}$$

if we denote $\boldsymbol{x}_1 \triangleq \boldsymbol{\mu}_{*,i}^t$, $\boldsymbol{x}_2 \triangleq \boldsymbol{\mu}_{*,i}^t$+**vec**$(\delta_j)$ where **vec**(δ_j) is a vector with all of its elements are zero except $\delta_j \geq 0$, $\forall \alpha \in [0,1]$ the difference of $\mathscr{V}(\boldsymbol{x}_1) = \alpha\psi_1(\boldsymbol{x}_1) + (1-\alpha)\psi_1(\boldsymbol{x}_2)$ and $\mathscr{V}(\boldsymbol{x}_2) = \psi_1(\alpha\boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2)$ can be represented with:

$$\begin{aligned}&\mathscr{V}(\boldsymbol{x}_1) - \mathscr{V}(\boldsymbol{x}_2) \\ &= \frac{\alpha(1-\alpha)(\delta_j\mu_{j,i}^t)^2}{(\mathbb{1}\cdot\boldsymbol{x}_1 - \sum_{j=1}^{N}\Lambda_{j,i}^t)(\mathbb{1}\cdot\boldsymbol{x}_1 - \sum_{j=1}^{N}\Lambda_{j,i}^t + \delta_j)}\end{aligned}$$

. As the constraint (43) ensures the non-negativity of $\mathbb{1}\cdot\boldsymbol{x}_1$ - $\sum_{j=1}^{n}\Lambda_{j,i}^t$, the difference of $\mathscr{V}(\boldsymbol{x}_1)$ and $\mathscr{V}(\boldsymbol{x}_2)$ will be always be non-negative. Therefore, for any $\boldsymbol{x} \leq \boldsymbol{y}$, we can construct a series of intermediate vectors in the form of

$$\boldsymbol{x}^{(z)} = \quad \boldsymbol{x}^{(z-1)} + \mathbf{vec}(\delta_z) \tag{52}$$

where $\boldsymbol{x}^{(0)} = \boldsymbol{x}$ and $\boldsymbol{x}^{(n)} = \boldsymbol{y}$. Therefore, we can get

$$\mathscr{V}(\boldsymbol{x}) - \mathscr{V}(\boldsymbol{y}) = \sum_{z=1}^{n}\left(\mathscr{V}(\boldsymbol{x}^{z-1}) - \mathscr{V}(\boldsymbol{x}^z)\right) \geq 0 \tag{53}$$

which guarantees the convexity of ψ_1 in its feasible region. Besides, the Constraints (11), (12), (43) can also be summarized with a linear system. Thus, $\mathcal{P}_2^{ORA}$ can be solved by existing and mature nonlinear optimization algorithms like the Generalized Reduced Gradient (GRG) method described in [34] which are implemented in commercial solvers [35], like LINGO[6] and MOSEK.[7] It is worth noting that constraint Equation (10) is replaced with Equation (43) in $\mathcal{P}_2^{RA}$, because we can easily prove that they are equivalent when we reconstruct Equation (42) with $\kappa_i^t = \gamma_{k,i}^t$ - $\lambda_{k,i}^t$.

Optimal Request Routing (ORR). In this sub-problem, the optimal service scheduling strategy that guides how hosts should process requests of different services $\boldsymbol{\vartheta}^t$ can be obtained by solving the following optimization problem:

$$\mathcal{P}_2^{ORR}: \quad \min_{\boldsymbol{p}^t} \quad \psi_2(\boldsymbol{\vartheta}^t) \quad s.t. \quad (20), (21), (42) \tag{54}$$

[6]https://www.lindo.com/
[7]https://www.mosek.com/

$$\begin{aligned}\mathbb{E}_\ell^t(\boldsymbol{\vartheta}^t, \boldsymbol{\varrho}^t, \boldsymbol{\mu}^t) &= \frac{1}{\Lambda_U^t}\sum_{i,j,k}\vartheta_{i,j,k}^t\Lambda_{j,i}^t \cdot \left[\tau_{i,j,k} + \frac{1}{\gamma_{k,i}^t - \lambda_{k,i}^t}\right] \\ &= \frac{1}{\Lambda_U^t}\sum_{i,j,k}\vartheta_{i,j,k}^t\Lambda_{j,i}^t\tau_{i,j,k} + \frac{1}{\Lambda_U^t}\sum_{i,j,k}\frac{\vartheta_{i,j,k}^t\cdot\Lambda_{j,i}^t}{\gamma_{k,i}^t - \sum_{j'=1}^{N}\vartheta_{i,j',k}^t\Lambda_{j',i}^t} \\ &= \frac{1}{\Lambda_U^t}\sum_{i,j,k}\vartheta_{i,j,k}^t\Lambda_{j,i}^t\tau_{i,j,k} + \frac{1}{\Lambda_U^t}\sum_{i,k}\frac{\sum_{j'=1}^{N}\vartheta_{i,j',k}^t\cdot\Lambda_{j',i}^t}{\gamma_{k,i}^t - \sum_{j'=1}^{N}\vartheta_{i,j',k}^t\Lambda_{j',i}^t} \\ &\geq \frac{1}{\Lambda_U^t}\sum_{i,j,k}\vartheta_{i,j,k}^t\Lambda_{j,i}^t\tau_{i,j,k} + \frac{1}{\Lambda_U^t}\sum_{i}\frac{1}{\frac{\sum_{k=1}^{N}\gamma_{k,i}^t}{\sum_{j'=1}^{N}\vartheta_{i,j',k}^t\Lambda_{j',i}^t} - 1} \\ &\geq \frac{1}{\Lambda_U^t}\sum_{i,j,k}\vartheta_{i,j,k}^t\Lambda_{j,i}^t\tau_{i,j,k} + \frac{1}{\Lambda_U^t}\sum_{i}\frac{N\sum_{k=1}^{N}\Lambda_{k,i}^t}{\sum_{k=1}^{N}\gamma_{k,i}^t - \sum_{k=1}^{N}\Lambda_{k,i}^t}\end{aligned}$$

This sub-problem $\mathcal{P}_2^{ORR}$ can be transformed into a linear programming when the decision variable $\boldsymbol{\vartheta}^t$=$\{\vartheta_{i,j,k}^t\}$ is vectorized with the order of i, j and k. Then the constraint Equation (42) can be transformed to:

$$\begin{bmatrix} \Lambda_{1,i}^t & \dots & \Lambda_{n,i}^t & \dots & 0 & \dots & 0 \\ 0 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & \Lambda_{1,i}^t & \dots & \Lambda_{n,i}^t \end{bmatrix} \boldsymbol{\vartheta}_i^t = \begin{bmatrix} \gamma_{1,i}^t - \kappa_i^t \\ \gamma_{2,i}^t - \kappa_i^t \\ \vdots \\ \gamma_{n,i}^t - \kappa_i^t \end{bmatrix}$$

Here $\boldsymbol{\vartheta}_i^t$ means the elements related to $\vartheta_{i,*,*}^t$ in vectorized $\boldsymbol{\vartheta}^t$. Therefore, the $\mathcal{P}_2^{RR}$ problem will be solved by adopting the dual-simplex algorithm [36].

Algorithm 2 request Routing, Data Delivery and Resource Allocation Algorithm (RDC)

Input: the target EPS = $(\mathcal{G}, \mathcal{S}, \mathcal{U})$
Output: $\boldsymbol{\varrho}^t$: the data delivery strategy in t-th time slot
$\boldsymbol{\mu}^t$: the resource allocation strategy in t-th time slot
$\boldsymbol{\vartheta}^t$: the service routing strategy in t-th time slot

```
for every time slot t do
    ϱ^t ← solve P'_{NR→} and P'_{NR←} with simplex method
    μ^t ← solve P_2^{ORA} with GRG method
    ϑ^t ← solve P_2^{ORR} with simplex method
end for
```

VI. RDC With Network Partition

Though we've divided the original problem $\mathcal{P}_1$ into several easy problems so that better request routing, data delivery, and resource allocation strategies can be found in a not time-consuming way in the former sections, it requires a long waiting to obtain the paths between all edge server pairs, even when this computation can be executed offline in advance. Besides this, the increasing of possible paths will enlarge the scale of problems $\mathcal{P}'_{\overrightarrow{NR}}$ and $\mathcal{P}'_{\overleftarrow{NR}}$ as shown in Section V-B.1. In fact, except for some extreme scenarios, it is really not necessary to have too many candidate paths for data delivery and request routing in most edge computing application scenarios, because this multi-hop transmission will result in external latency and unreliability, which is against the original intention of the MEC architecture [37], [38]. Thus, it will be acceptable to consider the data delivery and routing paths within a limited hop number. This limitation will lead to the partition of the graph into several sub-graphs because a part of edge servers will become unreachable for other edge servers in this case. Therefore, when we successfully partition the origin edge graph $\mathcal{G}$ into K sub-graphs ($\mathcal{G}_1$, $\mathcal{G}_2$, …, $\mathcal{G}_K$), the proposed algorithms in former sections can be applied to them in parallel, so that the total computation complexity can nearly be reduced to at most $\frac{|\bar{V}_s|!}{N!} \cdot 100\%$ of the origin when $\bar{V}_s$ is the average edge server number of the sub-graphs. It will be obvious that in partitioning the original graph $\mathcal{G}$ into several appropriate sub-graphs, the key point is to find the cliques where the edge servers of the same clique can have good communication quality with each other. Therefore, we use $\mathscr{Q}_{j,k}$ to measure the communication quality between the edge servers, where it is quantified with

$$\mathscr{Q}_{j,k} = \zeta \cdot \mathscr{D}\langle h_j, h_k\rangle + \frac{\sum_{i=1}^{M}(\sum_{z=1}^{N}\Lambda_{z,i})(I_i + O_i)}{\Lambda_U \mathscr{B}\langle h_j, h_k\rangle} \quad (55)$$

Thus, when we use

$$\mathbf{M}(\sigma) = \frac{1}{2\Upsilon}\sum_{j,k}\left[\mathscr{Q}_{j,k} - \frac{\mathscr{N}_j\mathscr{N}_k}{2\Upsilon}\right]\sigma(h_j, h_k) \quad (56)$$

to describe the modularity [39] of the partition result where $\mathscr{N}_j = \sum_k \mathscr{Q}_{j,k}$ is the sum of the communication quality for edge server h_k in transmission, $\Upsilon = \frac{1}{2}\sum_{j,k}\mathscr{Q}_{j,k}$ is the global communication quality of the network and the value partition function $\sigma(h_j, h_k)$ is 1 if edge server h_j and h_k are in the same group otherwise 0. Obviously, the modularity is a scalar value in [-1, 1] when given a partition function $\sigma(\cdot)$ of a graph. What's more, we can also review the modularity expression from the perspective of the partition results derived by the partition function $\sigma(\cdot)$:

$$\begin{aligned}\mathbf{M}(\sigma) =& \frac{1}{2\Upsilon}\left[\sum_{j,k}\sigma(h_j,h_k)\mathscr{Q}_{j,k} - \frac{\sum_{j,k}\mathscr{N}_j\mathscr{N}_k\sigma(h_j,h_k)}{2\Upsilon}\right] \\ =& \frac{1}{2\Upsilon}\left[\sum_{G'}\left(\sum_{h_j,h_k\in G'}\mathscr{Q}_{j,k}\right) - \sum_{G'}\frac{\left(\sum_{h_j\in G'}\mathscr{N}_j\right)\left(\sum_{h_k\in G'}\mathscr{N}_k\right)}{2\Upsilon}\right] \\ =& \sum_{G'}\left[\frac{\sum_{h_j,h_k\in G'}\mathscr{Q}_{j,k}}{2\Upsilon} - \left(\frac{\sum_{h_z\in G'}\mathscr{N}_z}{2\Upsilon}\right)^2\right]\end{aligned}$$

where $\frac{\sum_{h_j,h_k\in G'}\mathscr{Q}_{j,k}}{2\Upsilon}$ measures the communication quality among edge servers inside group G' and $\left(\frac{\sum_{h_z\in G'}\mathscr{N}_z}{2\Upsilon}\right)^2$ measures the communication quality of the edge servers inside and outside of group G', then the modularity can measure the density of interactions inside groups as compared to interactions between groups — if we can maximize the modularity, the groups represented with σ^* that have high internal communication qualities will be found:

$$\sigma^* = \underset{\sigma}{\operatorname{argmax}}\,\mathbf{M}(\sigma), \quad (57)$$

and we follow the approach of Blondel et al. [40], [41] to achieve this goal for its outstanding $O(n\log n)$ time complexity by checking and maximizing the modularity gain $\Delta\mathbf{M}(\sigma)$ of adding edge server h_z into group G':

$$\begin{aligned}\Delta\mathbf{M}(\sigma) =& \left[\frac{\sum_{h_j\in G'}\mathscr{Q}_{z,j} + \sum_{h_j,h_k\in G'}\mathscr{Q}_{j,k}}{2\Upsilon} - \left(\frac{\mathscr{N}_z + \sum_{h_j\in G'}\mathscr{N}_j}{2\Upsilon}\right)^2\right] - \left[\frac{\sum_{h_j,h_k\in G'}\mathscr{Q}_{j,k}}{2\Upsilon} - \left(\frac{\sum_{h_j\in G'}\mathscr{N}_j}{2\Upsilon}\right)^2 + 0 - \left(\frac{\mathscr{N}_z}{2\Upsilon}\right)^2\right] \\ =& \frac{1}{2\Upsilon}\left[\sum_{h_j\in G'}\mathscr{Q}_{z,j} + \frac{\mathscr{N}_z\sum_{h_j\in G'}\mathscr{N}_j}{\Upsilon}\right]\end{aligned}$$

Based on this idea, the original graph of the EPS $\mathcal{G}$ can be divided into several clusters by greedily integrating the edge servers that maximize $\Delta\mathbf{M}(\sigma)$ in an iterative way, so that edge servers of the same cluster can interact with each other within

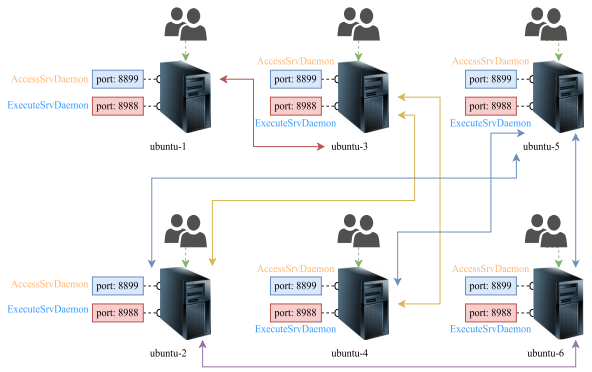

Fig. 5. An example of constructing a VM-based MEC service system.

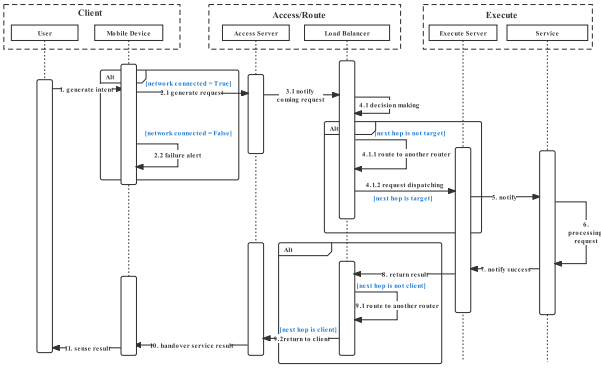

Fig. 6. The sequence diagram to show the interactions in the system.

a small number of hops, and then the proposed RDC algorithm can have its parallel version, whose name is RDC-NeP, on these individual clusters.

VII. Experiments and Analysis

A. Preliminary

Owing to the absence of well-defined platforms and fully developed datasets, this study employs a synthetic approach to generate experimental data. Prior to commencing the numerical experiments, it is imperative to ascertain the validity of our model. To this end, we establish a micro-MEC environment utilizing virtual machines as edge servers and regulate the communication quality using bandwidth-control tools. By collecting the time costs of requests in the micro-MEC system and comparing their average with the theoretical model, we can verify the reliability of the theoretical model — the rationality of the theoretical model will allow us to run the following numeric experiments of comparing our solution with some representative baselines. Fig.5 shows the components that the micro-MEC environment is constructed with (mainly powered by `java.nio.*`), and Fig.6 is the sequence diagram that describes the interactions among these components.

Fig.7 is a violin plot[8] to show the time cost distribution of different service requests in the created micro-MEC environment. In this figure, the dark dot stands for the average time cost of the service requests and the dark solid line shows the median value of the time costs. In summary, the result shows that the average response time of the requests in the micro-MEC environment is ∼1.354s while the theoretical

[8]https://ww2.mathworks.cn/matlabcentral/fileexchange/91790-al_goodplot-boxblot-violin-plot

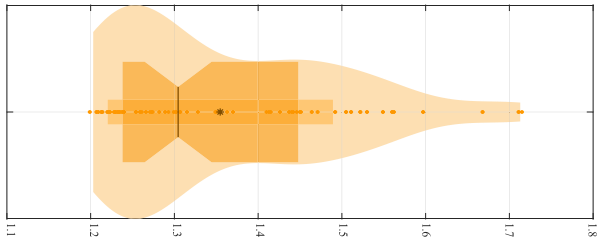

Fig. 7. The violin plot of the response time distribution in the micro-MEC environment and a normal distribution.

model obtains ∼1.296s. With less than 4.28% error in estimating the average response time, it will be tolerated to accept the appropriateness of our theoretical models and perform further analysis based on the numerical experiment results.

In the following sections, the comparison experiments are conducted first to check whether the proposed method can outperform other baselines and whether the heuristic pre-processing in network partition can help to accelerate the computation without much optimum loss. Besides, a series of parameter experiments are also conducted to check the impacts of different MEC-based service system settings.

B. Baselines

Because the RDC and RDC-NeP algorithm is the first attempt to solve this particular problem in an edge-cloud cooperation service provisioning system, existing algorithms are not suitable to be applied directly here. Therefore, we select some popular and representative approaches which aim at solving similar problems in this section as baselines:

1) *Random Strategy (RS) [42].* As used in our former work, the random strategy *RS* is the goalkeeper of all the possible approaches. It means no external knowledge is taken advantage of to solve the problem, where all the decision variables are randomly determined (but constraints will hold).

2) *Frequency-based Strategy (FBS) [43].* In this approach, *FBS* will allocate resource according to the service frequency so that the most frequently used services will have the most resource and requests.

3) *Workload Sensitive Strategy (WSS) [44].* In the *WSS* approach, it is necessary for the hosts to improve resource utilization based on the actual workload needs of services. When the workload is increased or reduced, some service resource will be added or released accordingly so that the heaviest services will have the most resource and requests.

4) *Adaptive Popularity Strategy (APS) [45].* The *APS* can be regarded as a dynamic version of *FBS* where some details may vary. In this approach, the popularity is not totally determined by the current frequency but also considers the contributions of historical service frequency. It will allocate resources according to the dynamically updated popularity of services so that the most popular services will have the most resource and requests.

To make the comparison more persuasive, the evaluation of these approaches is conducted with the settings in Table. I in which the settings are reasonable in the real world. It is worth noting that though the paper extends the existing work in [10], the proposed PCA-CATS strategy in that is not suitable to be selected in the comparison experiments. This is because the selected baselines should have the ability to determine how the data is delivered, how the resource is allocated, and how the request is routed. However, the PCA-CATS cannot generate a correct data delivery strategy.

TABLE I
SYSTEM CONFIGURATIONS

Item	Value	Item	Value
M	3	N	10
v_j	$\mathcal{N}(2{,}0.2^2)$ MB/s	$B_{j,k}$	$\mathcal{N}(100{,}10^2)$ MB/s
d_i^I	$\mathcal{N}(2{,}0.3^2)$ MB	d_i^O	$\mathcal{N}(2{,}0.3^2)$ MB
η_j	$\mathcal{N}(10{,}1^2)$ \$/MIPS	μ_j	$\mathcal{N}(6{,}1^2)\times 10^2$ MIPS
$\Lambda_{j,i}^t$	$\mathcal{N}(10,\ 1^2)$ QPS	w_i	$\mathcal{N}(10{,}2^2)\times$ MI

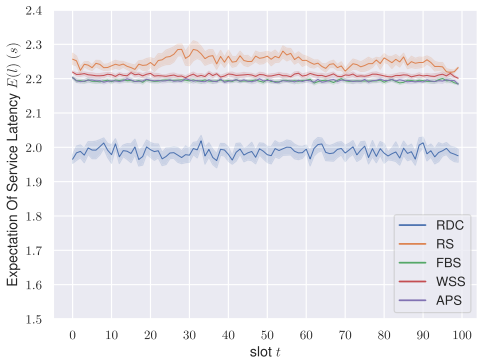

Fig. 8. The average service request latency of different methods.

C. Comparison With Baseline Algorithms

From the average latency curves shown in Fig.8, we can see that RDC has a significant advantage over the baseline methods when the system configurations are set equal for these approaches, but the differences among the baseline methods are relatively small. This result shows that our approach can better utilize the given resources to achieve higher performance while the baselines may be not good at it. Though the final comparison results highlight the improvement of the RDC, it is not suitable to deny the practicalities of the baselines. Actually, since the baselines are easy to implement and show generality for similar problems, it will be a good trial to apply them to start the system. Among the baselines, we can also see that the *APS* and *FBS* have some advantages in terms of performance compared to the *WSS* and *RS*. At the same time, it is worth noting that the difference between *APS* and *FBS* in performance is hard to discern in the figure when all curves are illustrated on it. This may be due to the fact that the *APS* and *FBS* are also able to adjust the resource allocation dynamically, like ours, according to the changes in request traffic.

As only observing the performance of the methods from the perspective of average latency cannot exhibit the comparison in a multifaceted way, here we draw Fig.9 as well to illustrate the detailed differences among the proposed approach RDC and the baselines from multi-dimension perspectives. To generate Fig.9, we randomly change the set of average service request arrival rate $\{\Lambda_{i,j}\}$ to simulate the different request patterns, and then apply RDC and other baselines on these independent sets to obtain the pair of average service request expense and latency with a big enough T=200. When all the pairs of (average expense, average service request latency) are collected and drawn on the expense-latency coordinate, we further count their frequency and visualize the distribution with a heat-map in Fig.9(c), Fig.9(d), Fig.9(e), Fig.9(f) then, while the histograms of these two types of values are also shown individually in Fig.9(a), Fig.9(b). From these sub-figures, we can clearly see the advantage of RDC in performance-cost optimization because the strategies generated by it will always achieve a better performance and a better expense at the same time while the other baselines are not able to.

TABLE II
DPM AND RDC RUNNING TIMES BEFORE AND AFTER PARTITION

Graph	DPM time (s)	RDC time (s)
origin	$1.99566s$	6.74
sub_0	1.99×10^{-3}	0.09
sub_1	1.98×10^{-3}	0.09
sub_2	2.99×10^{-3}	0.20
sub_3	1.97×10^{-3}	0.09
sub_4	1.03×10^{-3}	0.08
sub_5	1.03×10^{-3}	0.09
sub_6	2.03×10^{-3}	0.10

D. Impact of Network Partition

A further comparison lies in the application of the network partition, or the NeP module. In order to illustrate the effectiveness of RDC-NeP method, we first apply the NeP algorithm on an MEC-based service system with 50 edge nodes and partition it into multiple sub-graphs (see in Fig. 10, the sub-graphs are labeled with different colors). Then, the average service request delay of each sub-graph and the original graph will be solved by the RDC algorithm. At the same time, we will also record the time cost in DPM when enumerating the delivery paths of each sub-graph and that of the original graph to check how much computation time it will reduce after using the NeP algorithm.

As shown in Fig. 11, the average service request latency curve for RDC-NeP stands higher than that of RDC, and this slight gap, around 6% advantage, tends to be stable after several time slots. This is because that the RDC algorithm can have a global overview of the system information and make use of all the resources to make final decisions, while some requests may not be routed to the best server for processing with the RDC-NeP algorithm. But the earning of using RDC-NeP will still please us. As shown in Table II RDC-NeP has an overwhelming advantage over RDC in the total time cost, it enables the dynamical adjusting for the MEC-based service system, especially when solving the DPM problem for a complex and volatile network topology is a great deal. In conclusion, RDC-NeP method can provide a good solution when the problem scale is large, and greatly reduce the time consumption of solving DPM problem without an unacceptable loss of the final performance.

E. Impact of System Configurations

To explore how the system configurations will affect the resource allocation and request routing strategies under the RDC algorithm, we try to apply the RDC algorithm to groups of different system configurations in this section.

1) Impact of Service: First, we will check the impacts of services, including the data size of input and output, as well as the workloads.

1) Impact of input and output data size: In Fig. 12(a), we show how the expectation of service latency will change with the increase of input and output data size. Obviously, with the increase in the data size, the average service request time shows will also increase. This is because with the increase of input and output data size, the value of $\tau_{i,j,k}$ that is defined with $\tau_{i,j,k} = (I_i + O_i)/\nu_j + F_{i,j,k}$ will be enlarged, and this

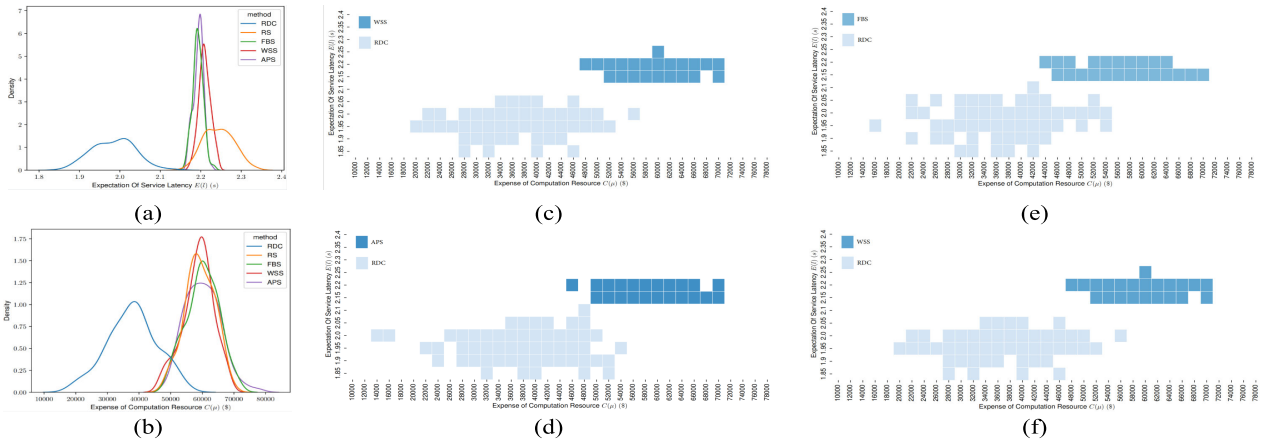

Fig. 9. (a) The performance distributions with different request flows (b) The expense distributions with different request flows (c) The performance-expense distribution heat-map of WSS vs. RDC (d) The performance-expense distribution heat-map of APS vs. RDC (e) The performance-expense distribution heat-map of FBS vs. RDC (f) The performance-expense distribution heat-map of WSS vs. RDC.

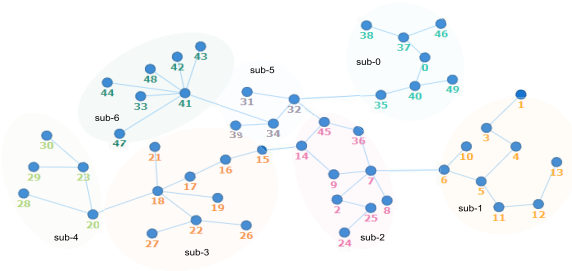

Fig. 10. A test case of a 50-edge server system after network partition.

Fig. 11. The average service request latency of RDC and RDC-NeP.

will finally result in the increase of average request delay. It is still worth noting that the impacts of $\bar{I}$ and $\bar{O}$ are really close in these figures, this is because I_i and O_i are symmetric in Equation (31).

2) Impact of workload: The workload of service measures the effort that the executor needs to make on processing this service. In Fig.12(b), we have shown how the expectation of service latency will change with the increase of service workload. In this figure, it can be observed that as the workload increases, the average request latency shows an upward trend. This is because the increase in workload will result in a longer time in request execution, and the increase of $\ell^E_{i,k}$ in Equation (17) will rise up the total latency.

2) Impact of Edge Server: Secondly, we will check the impacts of edge servers, including the arrival rate of service requests they have received and the available resource they have.

1) Impact of arrival rate: After all, the service requests can be seen as inputs if we regard the MEC-based service system as a black box, and large request arrival rates stand for frequent interactions within the system. The experiment about service request arrival rate will evaluate the flexibility of the system. From Fig. 12(c), we can find that the expected time cost rises steadily with the increase of service request arrival rate. The busily working IoT devices will definitely result in burdens and reduce performance. Like typical queue systems, the total time cost here does not increase linearly — the increase from 10 qps to 15 qps is higher than that from 5 qps to 10 qps in this experiment. If a higher requirement of arrival rate is claimed by the users, it will be suggested to scale up or scale out the original system.

2) Impact of available resource: The available resource of an edge server measures the capacity of this edge server in handling service requests. In Fig. 12(d), we show how the expectation of service latency will change with the increase of the average available resources. It can be found that as $\mu^\star$ increases (from 300 to 400 MIPS), the average request latency shows a downward trend. This is because, with more available resource provided, more resource can be allocated to individual service instances so that the processing of individual service requests can be completed more quickly. Formally, we can extract the related term from the latency estimation expression in Equation (24) to analyze how the available resource will affect the result:

$$\begin{aligned}
&\left(\sum_{i,j,k}\frac{w_i\Lambda_{i,j}\vartheta_{i,j,k}}{\mu_{k,i}-w_i\lambda_{k,i}}\right)^{-1}\\
&\quad\leq\quad\frac{\sum_{j,k}(\sum_i\frac{\mu_{k,i}}{w_i\Lambda_{i,j}\vartheta_{i,j,k}}-\sum_i\frac{\lambda_{k,i}}{\Lambda_{i,j}\vartheta_{i,j,k}})}{M^2}\\
&\quad\leq\quad\frac{1}{M^2}\sum_{j,k}\sum_i\frac{\mu_{k,i}}{w_i\Lambda_{i,j}\vartheta_{i,j,k}}\\
&\quad\leq\quad\frac{1}{M^2w_\circ\Lambda_U}\sum_{j,k}\sum_i\mu_{k,i}\leq\frac{1}{M^2w_\circ\Lambda_U}\sum_{j,k}\mu^\star_k\\
&\quad\Rightarrow\quad\frac{M^2w_\circ\Lambda_U}{\sum_{j,k}\mu^\star_k}\leq\left(\sum_{i,j,k}\frac{w_i\Lambda_{i,j}\vartheta_{i,j,k}}{\mu_{k,i}-w_i\lambda_{k,i}}\right)
\end{aligned}\tag{58}$$

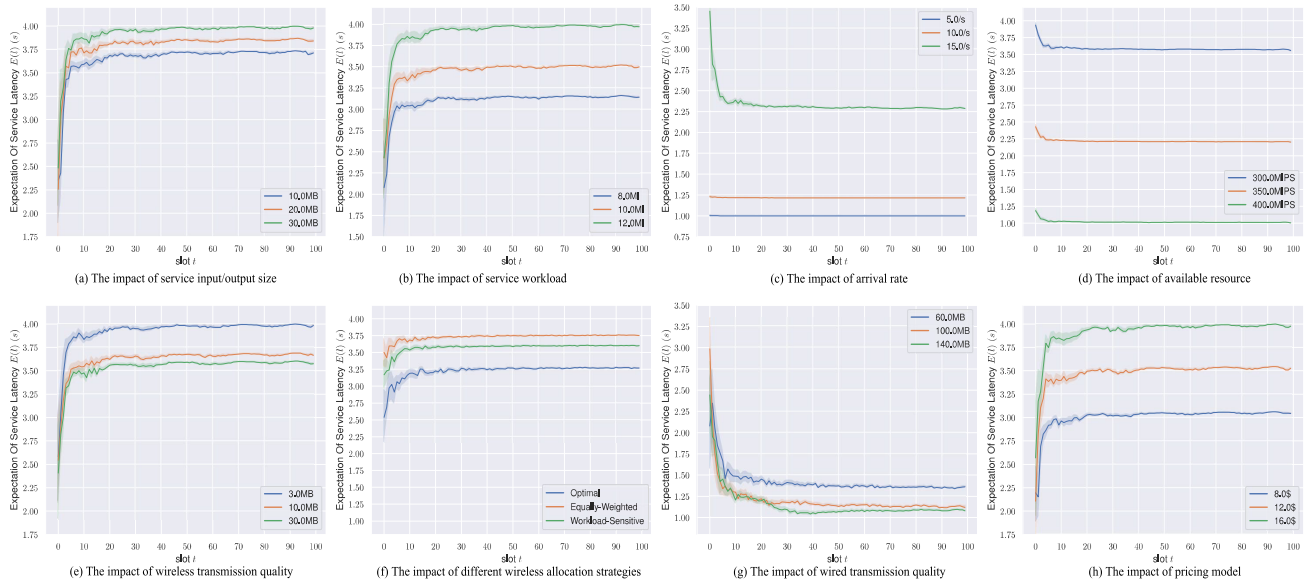

Fig. 12. Impact of system configurations.

With the AM-HM inequality and $w_\circ$ representing the minimum workload of the services, it can be found that the increase of $\mu_k^\star$ will reduce the lower bound of the expected total time cost for a service request, which means the increasing of available resource on edge servers can provide more room for the RDC to optimize the performance.

3) Impact of Network: **1) Impact of wireless transmission rate:** The wireless transmission rate ν_j measures how fast the mobile devices can communicate with the edge server h_j. In Fig. 12(e), we have shown how the expectation of request time will change with the increasing of wireless transmission rate ν_j. It can be found that as the wireless transmission rate increases, the average request latency shows a downward trend. This decline is mainly due to the fact that the increase of ν_j makes the value of $\tau_{i,j,k}$, which is defined with $\tau_{i,j,k} = (I_i + O_i)/\nu_j + F_{i,j,k}$, be smaller. And the value of $\tau_{i,j,k}$ will further reduce the final result in Equation (24). In this situation, the average wireless transmission rate ν_j is homogeneous for all the mobile devices for its meaning of "average" because the wireless bandwidth allocation is usually not managed by the EPS for its scope of responsibility. However, if we can break the barrier between the facility providers (e.g. the operators of h_1-h_N) and the service providers (e.g. the operators of s_1-s_M), then we can do more in dynamical wireless bandwidth allocation. In this case, the related problem is:

$$\mathcal{P}_\nu: \quad \min_{\nu_{i,j}} \quad \frac{1}{\Lambda_U}\sum_{j=1}^{N}\sum_{i=1}^{M}\frac{\Lambda_{i,j}(I_i+O_i)}{\nu_{i,j}}$$
$$s.t. \sum_{i=1}^{M}\nu_{i,j} \le \nu_j, \tag{59}$$

if $\nu_{i,j}$ is the wireless bandwidth allocated to service s_i on h_j and ν_j becomes the total wireless resource of h_j. The reason that this problem can be picked out from the original problem is that other items are not related to $\nu_{i,j}$. Obviously, the optimal wireless bandwidth allocation strategy $\{\nu_{i,j}\}$ can be easily generated by $\nu_{i,j} = \frac{\sqrt{\Lambda_{i,j}(I_i+O_i)}}{\sum^M{}_{z=1}\sqrt{\Lambda_{z,j}(I_z+O_z)}} \cdot \nu_j$ with the Lagrange multiplier method. The comparisons are shown in Fig.12(f), we can clearly find the differences after the EPS becomes able to take charge of the wireless management — the rearranged wireless bandwidth allocation strategies outperform the original one which treats every service equally. What's more, the wireless bandwidth allocation strategy generated by solving problem $\mathcal{P}_\nu$ shows the best result in reducing the final latency.

2) Impact of wired transmission rate: The wired transmission rate is $\mathscr{B}$ measures how fast the edge servers can communicate with each other, namely, the communication quality or cooperation efficiency in the edge cluster. In Fig 12(g), we use three curves to show how the expectation of request time will change with the increasing of wired transmission rate. It can be found that as the wired transmission rate increases, the average request time shows a downward trend. Similarly, this decline is mainly due to the fact that the increase of $\mathscr{B}$ makes τ smaller.

4) Impact of Pricing Model: As depicted in Fig. 12(h), the expectation of request time exhibits an upward trend with the increase in resource prices. This increase in latency can be attributed to the fact that the final cost is proportional to the price η. Given a fixed total cost ceiling C_{max}, the actual resources allocated to each service decrease, resulting in an increase in the average service request time.

VIII. Conclusion and Future Work

In conclusion, this study introduces a novel approach for determining an approximate optimal service provisioning strategy in an MEC environment with respect to resource allocation, traffic scheduling, and data delivery. Utilizing the Lyapunov optimization technique, we develop an online service provisioning strategy analysis framework that dynamically adapts the system to achieve optimal performance within acceptable costs. Experimental results demonstrate that our approach, referred to as RDC, surpasses other baseline methods while its variant, RDC−NeP, also exhibits commendable performance on large-scale networks However, there are still some points that we do not reach:

1) Services do not play alone in many scenarios. This means that it is not enough to consider the cooperation of edge servers or clouds, the cooperation (or composition) of services [46] is also of great importance.
2) Services may be redundant in practice. As software-as-a-service technology and the market evolve, there will be many services with the same functionality but different qualities [47]. We need to consider the choices among them.
3) Strategies may be more simple. As shown in the former sections, though the targeted problem can be decomposed into several convex optimization sub-problems, the problem-solving may also be time-consuming. Therefore, it is necessary to find whether there were some simple but practical approaches in strategy generation or to train some learning-based strategy models offline.

In the future, we would like to go further by investigating how these listed factors will affect service provisioning in the MEC environment.

ACKNOWLEDGMENT

Author Contributions: Zhengzhe Xiang analyzed the problem and proposed the algorithm and analyzed the experiment results; Yuhang Zheng was responsible for the experiments, while Zhengzhe Xiang and Zengwei Zheng also contributed to the plotting; and Shuiguang Deng, Minyi Guo, and Schahram Dustdar contributed to the modeling and formulation.

REFERENCES

[1] Y. Guo, R. Zhao, S. Lai, L. Fan, X. Lei, and G. K. Karagiannidis, "Distributed machine learning for multiuser mobile edge computing systems," *IEEE J. Sel. Topics Signal Process.*, vol. 16, no. 3, pp. 460–473, Apr. 2022.

[2] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5G mobile edge computing: Architectures, applications, and technical aspects," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1160–1192, 2nd Quart., 2021.

[3] Z. Chang, S. Liu, X. Xiong, Z. Cai, and G. Tu, "A survey of recent advances in edge-computing-powered artificial intelligence of things," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 13849–13875, Sep. 2021.

[4] Z. Xiang, S. Deng, Y. Zheng, D. Wang, J. Tehari, and Z. Zheng, "Energy-effective IoT services in balanced edge-cloud collaboration systems," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Sep. 2021, pp. 219–229.

[5] A. Shahidinejad, F. Farahbakhsh, M. Ghobaei-Arani, M. H. Malik, and T. Anwar, "Context-aware multi-user offloading in mobile edge computing: A federated learning-based approach," *J. Grid Comput.*, vol. 19, no. 2, pp. 1–23, Jun. 2021.

[6] 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.

[7] T. Bahreini, H. Badri, and D. Grosu, "Mechanisms for resource allocation and pricing in mobile edge computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 667–682, Mar. 2022.

[8] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. Shen, "Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 968–980, Jul./Aug. 2021.

[9] P. Lai et al., "Cost-effective app user allocation in an edge computing environment," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1701–1713, Jul./Sep. 2022.

[10] Z. Xiang, S. Deng, F. Jiang, H. Gao, J. Tehari, and J. Yin, "Computing power allocation and traffic scheduling for edge service provisioning," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Oct. 2020, pp. 394–403.

[11] B. Brik, P. A. Frangoudis, and A. Ksentini, "Service-oriented MEC applications placement in a federated edge cloud architecture," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

[12] A. Moubayed, A. Shami, P. Heidari, A. Larabi, and R. Brunner, "Edge-enabled V2X service placement for intelligent transportation systems," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1380–1392, Apr. 2020.

[13] L. Liu, M. Zhao, M. Yu, M. A. Jan, D. Lan, and A. Taherkordi, "Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 2169–2182, Feb. 2023.

[14] S. Bi, L. Huang, and Y.-J.-A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4947–4963, Jul. 2020.

[15] 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.

[16] A. Mohajer, M. S. Daliri, A. Mirzaei, A. Ziaeddini, M. Nabipour, and M. Bavaghar, "Heterogeneous computational resource allocation for NOMA: Toward green mobile edge-computing systems," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 1225–1238, Mar./Apr. 2023.

[17] 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.

[18] Y. Liu, X. Shang, and Y. Yang, "Joint SFC deployment and resource management in heterogeneous edge for latency minimization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 8, pp. 2131–2143, Aug. 2021.

[19] N. Akhtar, I. Matta, A. Raza, L. Goratti, T. Braun, and F. Esposito, "Managing chains of application functions over multi-technology edge networks," *IEEE Trans. Netw. Serv. Manage.*, vol. 18, no. 1, pp. 511–525, Mar. 2021.

[20] M. Kim, M. Jaseemuddin, and A. Anpalagan, "Deep reinforcement learning based active queue management for IoT networks," *J. Netw. Syst. Manage.*, vol. 29, no. 3, pp. 1–28, Jul. 2021.

[21] M. Bukhsh, S. Abdullah, and I. S. Bajwa, "A decentralized edge computing latency-aware task management method with high availability for IoT applications," *IEEE Access*, vol. 9, pp. 138994–139008, 2021.

[22] K. Ergun, R. Ayoub, P. Mercati, D. Liu, and T. Rosing, "Energy and QoS-aware dynamic reliability management of IoT edge computing systems," in *Proc. 26th Asia South Pacific Design Autom. Conf.*, Jan. 2021, pp. 561–567.

[23] Y. C. Lee, C. Wang, A. Y. Zomaya, and B. B. Zhou, "Profit-driven service request scheduling in clouds," in *Proc. 10th IEEE/ACM Int. Conf. Cluster, Cloud Grid Comput.*, May 2010, pp. 15–24.

[24] J. Vilaplana, F. Solsona, I. Teixidó, J. Mateo, F. Abella, and J. Rius, "A queuing theory model for cloud computing," *J. Supercomput.*, vol. 69, no. 1, pp. 492–507, Jul. 2014.

[25] M. Liu, Y. Mao, and S. Leng, "Cooperative fog-cloud computing enhanced by full-duplex communications," *IEEE Commun. Lett.*, vol. 22, no. 10, pp. 2044–2047, Oct. 2018.

[26] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, "*Q*-learning based dynamic task scheduling for energy-efficient cloud computing," *Future Gener. Comput. Syst.*, vol. 108, pp. 361–371, Jul. 2020.

[27] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Watch global, cache local: YouTube network traffic at a campus network: Measurements and implications," *Proc. SPIE*, vol. 6818, pp. 35–47, Jan. 2008.

[28] P. Burke, "The output process of a stationary M/M/s queueing system," *Ann. Math. Statist.*, vol. 39, no. 4, pp. 1144–1152, 1968.

[29] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[30] H. Zhao, S. Deng, C. Zhang, W. Du, Q. He, and J. Yin, "A mobility-aware cross-edge computation offloading framework for partitionable applications," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2019, pp. 193–200.

[31] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synth. Lectures Commun. Netw.*, vol. 3, no. 1 pp. 1–211, 2020.

[32] S. Kotera, B. Yin, K. Yamamoto, T. Nishio, M. Morikura, and H. Abeysekera, "Latency-aware fair scheduling for spatial reuse in WLANs: A Lyapunov optimization approach," in *Proc. IEEE 18th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2021, pp. 1–6.

[33] Y. Kawase, T. Matsui, and A. Miyauchi, "Additive approximation algorithms for modularity maximization," *J. Comput. Syst. Sci.*, vol. 117, pp. 182–201, May 2021.

[34] L. S. Lasdon, R. L. Fox, and M. W. Ratner, "Nonlinear optimization using the generalized reduced gradient method," *Revue Française d'automatique, Informatique, Recherche Opérationnelle. Recherche Opérationnelle*, vol. 8, no. 3, pp. 73–103, 1974.

[35] E. Goodarzi, M. Ziaei, and E. Z. Hosseinipour, *Introduction to Optimization Analysis in Hydrosystem Engineering*. New York, NY, USA: Springer, 2014.
[36] A. Koberstein, "Progress in the dual simplex algorithm for solving large scale LP problems: Techniques for a fast and stable implementation," *Comput. Optim. Appl.*, vol. 41, no. 2, pp. 185–204, Nov. 2008.
[37] K. Kaur, S. Garg, G. S. Aujla, N. Kumar, J. J. P. C. Rodrigues, and M. Guizani, "Edge computing in the industrial Internet of Things environment: Software-defined-networks-based edge-cloud interplay," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 44–51, Feb. 2018.
[38] H. Zhang, X. He, Q. Wu, and H. Dai, "Spectral graph theory based resource allocation for IRS-assisted multi-hop edge computing," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2021, pp. 1–6.
[39] T. Magelinski, M. Bartulovic, and K. M. Carley, "Measuring node contribution to community structure with modularity vitality," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 1, pp. 707–723, Jan. 2021.
[40] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Exp.*, vol. 2008, no. 10, Oct. 2008, Art. no. P10008.
[41] X. Su et al., "A comprehensive survey on community detection with deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 9, 2022, doi: 10.1109/TNNLS.2021.3137396.
[42] H. Zhao et al., "DPoS: Decentralized, privacy-preserving, and low-complexity online slicing for multi-tenant networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 12, pp. 4296–4309, Dec. 2022.
[43] G. Li, J. Wu, J. Li, K. Wang, and T. Ye, "Service popularity-based smart resources partitioning for fog computing-enabled industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4702–4711, Oct. 2018.
[44] C. Li, J. Liu, B. Lu, and Y. Luo, "Cost-aware automatic scaling and workload-aware replica management for edge-cloud environment," *J. Netw. Comput. Appl.*, vol. 180, Apr. 2021, Art. no. 103017.
[45] Z. Fan, W. Yang, F. Wu, J. Cao, and W. Shi, "Serving at the edge: An edge computing service architecture based on ICN," *ACM Trans. Internet Technol.*, vol. 22, no. 1, pp. 1–27, Feb. 2022.
[46] A. Palade and S. Clarke, "Collaborative agent communities for resilient service composition in mobile environments," *IEEE Trans. Serv. Comput.*, vol. 15, no. 2, pp. 876–890, Mar. 2022.
[47] A. Hussain, J. Chun, and M. Khan, "A novel customer-centric methodology for optimal service selection (MOSS) in a cloud environment," *Future Gener. Comput. Syst.*, vol. 105, pp. 562–580, Apr. 2020.

Zhengzhe Xiang (Member, IEEE) received the B.S. and Ph.D. degrees in computer science and technology from Zhejiang University, Hangzhou, China. He was a Visiting Scholar with Shanghai Jiao Tong University, Shanghai, China, in 2022. He is currently a Lecturer with the School of Computer and Computing Science, Hangzhou City University, Hangzhou. His research interests include service computing and edge computing. He serves as a Reviewer for several international journals, such as IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON SERVICES COMPUTING, *IET Communications*, and *Digital Communications and Networks*. He also serves as a PC member for many international conferences.

Yuhang Zheng is currently pursuing the M.S. degree with the College of Computer Science and Technology, Zhejiang University, Hangzhou, China. His research interests include the Internet of Things technology, edge computing, service computing, and reinforcement learning.

Zengwei Zheng received the B.S. and M.Ec. degrees in computer science and western economics from Hangzhou University, China, and the Ph.D. degree in computer science and technology from Zhejiang University, China, in 2005. He is currently a Full Professor with the School of Computer and Computing Science, Hangzhou City University, the Director of Intelligent Plant Factory, Zhejiang Province Engineering Laboratory, and the Director of the Hangzhou Key Laboratory for IoT Technology and Application.

Shuiguang Deng (Senior Member, IEEE) received the B.S. and Ph.D. degrees in computer science from Zhejiang University, China, in 2002 and 2007, respectively. He is currently a Full Professor with the College of Computer Science and Technology, Zhejiang University. Previously, he was a Visiting Scholar with the Massachusetts Institute of Technology in 2014 and Stanford University in 2015. His research interests include edge computing, service computing, cloud computing, and business process management. He is a fellow of IET. He serves as an Associate Editor for the journals, such as IEEE TRANSACTIONS ON SERVICES COMPUTING, *Knowledge and Information Systems*, *Computing*, and *IET Cyber-Physical Systems: Theory & Applications*. In 2018, he was granted the Rising Star Award by IEEE TCSVC.

Minyi Guo (Fellow, IEEE) received the Ph.D. degree in computer science from the University of Tsukuba, Japan. He is currently a Zhiyuan Chair Professor and the Head of the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. His current research interests include parallel/distributed computing, compiler optimizations, embedded systems, pervasive computing, big data, and cloud computing. He is also on the Editorial Board of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON CLOUD COMPUTING, and *Journal of Parallel and Distributed Computing*. He is a fellow of CCF.

Schahram Dustdar (Fellow, IEEE) is currently a Full Professor in computer science and the Head of the Distributed Systems Group, TU Wien. His research interests include distributed systems, and complex and autonomic software systems. He has received the ACM Distinguished Scientist Award and the IBM Faculty Award. He is an Elected Member of Academia Europaea, where he is the Informatics Section Chairperson. He is the Editor-in-Chief of *Computing*, the Co-Editor-in-Chief of *ACM Transactions on the Internet of Things*, and an Associate Editor of *ACM Transactions on the Web*, *ACM Transactions on Internet Technology*, IEEE TRANSACTIONS ON CLOUD COMPUTING, and IEEE TRANSACTIONS ON SERVICES COMPUTING. He is also on the editorial boards of IEEE INTERNET COMPUTING and *IEEE Computer Magazine*.