Task Allocation with Geography-Context-Capacity Awareness in Distributed Burstable Billing Edge-Cloud Systems

Shihao Shen, *Graduate Student Member, IEEE,* Chenfei Gu, Yuanze Li, Chao Qiu, *Member, IEEE,* Xiaofei Wang, *Senior Member, IEEE,* Rui Tan, *Senior Member, IEEE,* Cheng Zhang, and Wenyu Wang

***Abstract*—The new real-time interactive services, such as virtual and augmented reality, demand significantly higher network bandwidth and quality, which the traditional centralized cloud struggles to meet. In addition, centralized optimization management becomes inefficient as the scale of the scene continues to expand. In response, edge cloud systems have emerged, but distributed geographic locations, burstable billing business models, and large numbers of servers in large-scale scenarios pose new challenges for resource management. In this paper, we propose *GeoCC*, a novel strategy to save bandwidth overhead in burstable billing edge cloud systems. *GeoCC* addresses challenges through a dual approach. First, a geography-aware graph construction and partitioning algorithm is used to organize server resources, and a large number of servers are reasonably divided into multiple server pools for parallel processing. Second, it introduces an enhanced burstable billing optimization mechanism that considers contextual factors and adaptive bandwidth capacity. Experiments based on real data from an edge cloud operator demonstrate the effectiveness of *GeoCC*. Compared with the baseline, *GeoCC* can effectively reduce bandwidth peaks, decreasing bandwidth costs by an average of 28.30% and up to 81.83% at the 95th percentile billing.**

***Index Terms*—Edge-cloud architecture, bandwidth optimization, burstable billing, server pools partition.**

I. Introduction

In this section, we introduce the research background and 95th percentile billing, then discuss the research motivation and summarize the main contributions.

A. Background

In the ever-evolving landscape of digital connectivity, real-time interactive content services have emerged as the transformative force shaping the next generation of the Internet [2]–[4]. These trends highlight the pivotal role of real-time interactive content services in influencing the future of the Internet, permeating various industries and sectors. However, the evolution of real-time interactive content services is hindered by the limitations of the current centralized cloud computing paradigm. Traditional centralized cloud architectures face challenges in providing Quality of Service (QoS) for real-time interactive content services due to bandwidth constraints and the significant average network latency and jitter resulting from the separation of users and computing devices [5], [6]. While cloud computing has achieved great success, it is no longer a one-size-fits-all solution in today's environment.

In this context, the paradigm of edge cloud architecture emerges as a promising alternative. By bringing computation closer to the data sources, edge cloud architecture mitigates the challenges posed by centralized cloud architectures. The deployment of next-generation network infrastructures further supports the vision of edge cloud computing, offering a more responsive and efficient framework for real-time interactive content services [7], [8].

In the context of real-time interactive content services, the demand for substantial bandwidth from the edge cloud architecture is particularly pronounced during the transmission of live streams in both uplink and downlink directions. Currently, burstable billing has emerged as the primary method for calculating bandwidth expenses in edge cloud systems [5], [9]. This flexible billing approach has gained widespread adoption due to its adaptability to varying bandwidth usage patterns and its ability to prevent unexpected usage spikes and the resulted exorbitant costs for users.

When employing burstable billing, the 95th percentile billing emerges as the prevalent mechanism for measuring and charging bandwidth usage [10], [11]. It involves periodic billing cycles, commonly categorized as daily or monthly billing. In the case of monthly 95th percentile billing, the process consists of:

- ***Data Sampling***: The system samples the amount of transmitted data at regular intervals, typically every five minutes. These samples are stored, with each representing the amount of transmitted data during that interval.
- ***Sorting and Discarding***: At the end of the billing period, all sampled data points collected are sorted based on the average transmission rate. The top 5% of data points are then discarded, mitigating the impact of short-lived peaks in bandwidth usage.

Some earlier results of this paper were published in IEEE Global Communications Conference (GLOBECOM) 2023. [1] *(Corresponding author: Xiaofei Wang.)* The authors extend the previous work by improving the expression of parameters, adding complexity analysis to the problem, discussing the theoretical analysis and adding fine-grained experiments.

Shihao Shen, Chenfei Gu, Yuanze Li, Chao Qiu and Xiaofei Wang are with the College of Intelligence and Computing, Tianjin University, Tianjin 300350, China (e-mail: {shenshihao, chenfeigu, liyuanze, chao.qiu, xiaofeiwang}@tju.edu.cn).

Rui Tan is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore (e-mail: tanrui@ntu.edu.sg).

Cheng Zhang is with the Institute of Technology, Tianjin University of Finance and Economics, Tianjin 300222, China (e-mail: zccode@gmail.com).

Wenyu Wang is with the PPIO Cloud Computing (Shanghai) Co., Ltd., Shanghai 200120, China (e-mail: wayne@pplabs.org).

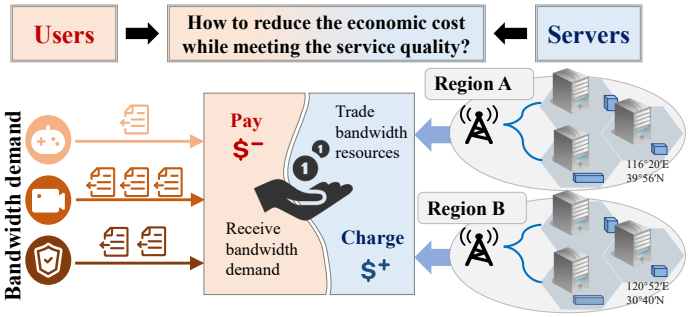

Fig. 1. Bandwidth demand allocation in edge-cloud system.

- ***Billing Calculation***: After discarding the top 5% of data points, the highest remaining bandwidth transmission rate determines the billed bandwidth usage for the billing period. The flexibility and adaptability of the 95th percentile billing makes it well-suited for the dynamic and varied nature of bandwidth usage.

B. Motivation

In commercial edge-cloud environments, the primary objective is to meet user demands with the lowest possible cost. This presents a critical challenge under the burstable billing model, where task allocation must be optimized to strike a balance between economic efficiency and operational performance. Consequently, designing an optimal allocation mechanism in such dynamic and cost-sensitive contexts remains an important and challenging problem. Based on a series of motivational experiments conducted with an edge-cloud operator[1], we observed a pronounced spatial and temporal heterogeneity in the supply and demand of computational resources [6]. These complexities necessitate the development of efficient task allocation strategies that can better align the supply of computational resources with user demand.

In the intricate landscape of edge cloud architectures, bandwidth billing usually also follows the above-mentioned burstable approach. This billing model, while offering flexibility, introduces new challenges that necessitate innovative scheduling approaches to optimize bandwidth expenditures [12]. Distinguishing itself from traditional Content Delivery Network (CDN), an edge cloud architecture encompasses both expansive server rooms and smaller, non-standardized server providers dispersed across diverse locations. This unique characteristic demands scheduling algorithms to factor in geographical considerations extensively, ensuring superior service quality. However, the proliferation of distributed nodes brings forth the challenge of proposing a unified scheduling mechanism that effectively addresses the intricacies of edge cloud architectures [13], [14]. The unique characteristics inherent to edge cloud architectures present potential challenges for scheduling, especially when considering economic efficiency, with three key aspects:

- **Geography Heterogeneity**: The varied distribution of servers across different geographic locations results in diverse bandwidth capabilities and pricing strategies. Failing to account for these geographical nuances in scheduling may lead to increased bandwidth expenses and diminished service quality [15], [16].
- **Context Directability**: The complex billing strategies, such as the 95th percentile billing, demand that current scheduling decisions rely on historical bandwidth usage data, introducing additional contextual information into each scheduling decision [17].
- **Capacity Exploitation**: The burstable billing nature of each server implies the existence of free bandwidth capacity. Maximizing the utilization of this capacity becomes imperative to reduce the costs associated with bandwidth expenses and fully exploit the available resources [18].

C. Main Contributions

In this paper, we present an innovative approach for optimizing bandwidth costs in edge cloud architectures, termed "*GeoCC*" - an abbreviation for our Geography-Context-Capacity aware task allocation strategy. *GeoCC* is devised to tackle the intricate challenge of reducing costs, as illustrated by the problem outlined in Fig. 1. *GeoCC* strategically integrates geographical location, context, and capacity awareness to achieve optimal cost efficiency. This paper represents a pioneering effort as proposing an allocation strategy that optimizes costs in edge-cloud systems featuring burstable billing, all without the need for prior knowledge of bandwidth demands. The distinctive contributions of this paper are detailed below.

- **Bandwidth Cost Optimization Model:** We design a model to optimize bandwidth overhead in edge-cloud scenarios, incorporating real-time allocation, server availability, user selectivity, non-linear burstable billing, and geographic dispersion constraints, and demonstrate that this problem is NP-Complete.
- **Geography-Aware Server Cluster Partitioning:** We propose a geography-aware graph construction strategy to build balanced server subsets, which addresses the spatial heterogeneity of server distribution and ensures equitable resource allocation.
- **Context and Capacity-Aware Burstable Billing Optimization:** We propose an enhanced burstable billing optimization mechanism based on balanced server subsets that integrates context awareness and adaptive bandwidth capacity to significantly improve cost efficiency in edge cloud environments.

II. System Framework

In this section, we first present the challenges faced in real-world scenarios. Next, we formalize the system model, capturing its key characteristics and constraints. Finally, we define the optimization objectives through precise mathematical formulations and analyze the complexity of the problem.

A. Scenario Problem Statement

In this scenario, three key roles are involved: the edge-cloud operator, the Internet Service Provider (ISP), and the

[1]PPIO Edge Cloud, PPIO Cloud Computing (Shanghai) Co., Ltd., https://www.ppio.cn

user. The edge-cloud operator (such as PPIO[2]) manages an edge cloud platform consisting of numerous geographically distributed edge servers. Similar to traditional cloud services, these edge servers rely on ISPs to access the internet infrastructure. ISPs employ the burstable billing model to charge the edge-cloud operator fees based on the bandwidth usage of the edge servers.[3] Concurrently, the edge-cloud operator hosts multiple services on these edge servers, capable of processing user request tasks to generate revenue. Therefore, we need to consider the optimization problem from the edge-cloud operator's perspective: how to allocate user tasks to minimize bandwidth costs under the burstable billing model while ensuring efficient task responses. It is also worth noting that this paper focuses on bandwidth cost optimization based on 95th percentile billing in bandwidth-intensive application scenarios, but resources such as computation and storage, which also incur costs, are outside the scope of the discussion.

Furthermore, due to the geographically distributed nature of edge computing, the edge-cloud operator faces more complex challenges compared with traditional centralized cloud systems. Take the PPIO mentioned above as an example, it operates more than 10,000 edge servers dispersed across more than 1,000 cities or regions, with the farthest distance between two edge servers exceeding 4,000 kilometers. Due to the excessive transmission distances and the large platform size, it is inefficient to use centralised optimisation for the global servers, so they need to be partitioned into multiple zones for autonomous management. Intuitively, using cities as the basis for partitioning edge servers seems straightforward. However, we find that the server resources in each city do not exactly match the user demand, i.e., there will be a large number of idle servers in some cities and insufficient servers in others. To achieve optimal performance, we must first partition the large-scale edge servers before addressing task allocation. Therefore, in Sec. II-C, we decompose this issue into two sub-problems: (i) Geography-aware server cluster partition and (ii) context-capacity-aware task allocation.

B. System Member Description

The proposed system, *GeoCC*, assumes the presence of P users, N real-time interaction content services, and M servers, denoted as $\mathcal{P}$, $\mathcal{N}$, and $\mathcal{M}$ respectively, and indexed by $p \in \{1,\ldots,P\}$, $n \in \{1,\ldots,N\}$, and $m \in \{1,\ldots,M\}$. *GeoCC* operates within discrete time intervals, indexed as $t \in \{1,\ldots,T\}$, denoted as $\mathcal{T}$. In order to improve the clarity of the model, the main parameters involved and their meanings are shown in Table I.

1) **User**: A user p submits a request denoted as $< T^p, N^p, B^p >$, where $T^p \in \{1,\ldots,T\}$ signifies the time interval when the request is generated. The content service type of the request is indicated by $N^p \in \{1,\ldots,N\}$, while the required bandwidth is denoted as B^p. In the edge cloud architecture, we believe that a user can submit multiple requests with different types in one time slot.
2) **Server**: The resource information for server m, represented as $< B_m, A_m >$, encompasses the available bandwidth resources, B_m, and the service availability of server m, A_m. Specifically, A_m is denoted as $A_m = a_m^1, a_m^2, \ldots, a_m^N$, where $a_m^n \in \{0,1\}$. When server m is capable of providing service for content service n, $a_m^n = 1$; otherwise, $a_m^n = 0$.
3) **Cluster**: In a real edge-cloud system, which may consist of thousands of servers, partitioning them into server clusters proved to be effective [19]. This enables the assignment of bandwidth requests to the nearest server cluster, enhancing the quality of time-sensitive services and reducing the scale of the allocation problem. Here, we assume the existence of J server clusters, indexed by $j \in \{1,\ldots,J\}$, denoted as $\mathcal{J}$. Additionally, we define $D_n^{t,j}$ as the cumulative bandwidth demand for service n requested by all users $p \in U_j$ with $T^p = t$. The expression is given by $D_n^{t,j} = \sum_{\forall p \in U_j, T^p = t, N^p = n} B^p$, where U_j represents the set of users assigned to server cluster j.

TABLE I
DEFINITION OF PARTLY CRITICAL PARAMETERS.

Parameters	Definitions
A_m	The set of service availability for server m.
a_m^n	The service availability of server m for service n.
B_m	The available bandwidth resources of server m.
B^p	The bandwidth required by the request generated by user p.
C_j	The set of servers within cluster j.
$D_n^{t,j}$	The sum of bandwidth demand for service n by all users belonging to server cluster j in time interval t.
$D_{n,m}^{t,j}$	The bandwidth demand of service n served by server m.
D'^j_m	The billed bandwidth of server m.
$\mathcal{D}_m^j$	A sequence of bandwidth usage records for servers m.
E	The edge set of the graph.
e_i^m	The edge connecting node m to node i.
$\mathcal{J}$	The set of server clusters.
j	Index of the server cluster.
M	The number of servers.
N	The number of real-time interactive content service types.
N^p	The content service type of the request generated by user p.
P	The number of users.
t	The Index of discrete time intervals.
T^p	The time interval for user p to generate a request.
$\mathcal{T}$	The set of discrete time intervals.
V	The point set of the graph.

C. System Description

GeoCC introduces a bandwidth cost economizing strategy, as illustrated in Fig. 2, comprising two main components:

1) Geography-Aware Server Cluster Partition: Upon completion of the partitioning, real-time interactive content requests from users will be redirected to the nearest bandwidth cluster. This partitioning, instead of considering the entire system as a whole, brings about two key advantages. Firstly, dividing the system into server clusters ensures that user traffic is only

[2]PPIO Edge Cloud, PPIO Cloud Computing (Shanghai) Co., Ltd., https://www.ppio.cn

[3]In real-world scenarios, link aggregation or clusters (multiple servers) sharing a single bandwidth may occur. However, for simplicity in this discussion, these are referred to as the server.

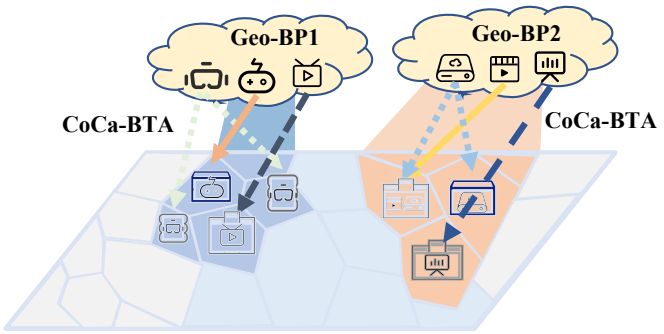

Fig. 2. The framework of *GeoCC*.

scheduled within nearby server clusters, preventing it from being directed to servers at a considerable distance and avoiding excessive latency, which could impact the quality of real-time interactive content services. Secondly, the division significantly reduces the complexity of subsequent request scheduling problems. Treating the whole edge cloud system as a cohesive entity comprising thousands of servers would greatly increase problem complexity, thereby diminishing the feasibility of designing online algorithms.

2) Context-Capacity-Aware Task Allocation: Following the aforementioned partitioning, we propose a context-capacity-aware task allocation algorithm that assigns servers within server cluster j to fulfill all requests allocated to the cluster.

Taking a request stream as an example, we assign the bandwidth demands $D_n^{t,j}$ to servers belonging to server cluster j and capable of providing service n. The bandwidth demand of a request flow may be satisfied by several different servers together, and the decision variable of the problem is to determine for each request flow the bandwidth resource of the service n provided by the server m, i.e., as $D_{n,m}^{t,j}$. Post-allocation, we have $D_n^{t,j} = \sum_{m \in C_j} D_{n,m}^{t,j}$, where C_j represents the set of servers within cluster j. Meanwhile, the sum of bandwidth demands undertaken by servers m must not exceed its available bandwidth resources. Moreover, our system promptly schedules requests within each server cluster to the corresponding servers, leveraging context and capacity awareness to capitalize on the burstable billing characteristic and reduce bandwidth costs.

This two-tier strategy, encompassing geography-aware server cluster partitioning and context-capacity-aware task allocation, forms the core of *GeoCC*, offering an efficient solution for optimizing bandwidth costs in real-time interactive services within edge-cloud systems.

D. Problem Formulation

We present our partition solution, denoted as $\widehat{\mathcal{P}}$. Our goal is how to partition server clusters from $\mathcal{J}$ such that each cluster is geographically centralized and relatively resource balanced, which is formally defined as:

$$\text{Minimize} \quad \max_{j \in \mathcal{J}} \max_{m_1, m_2 \in C_j} dist(m_1, m_2), \forall j \in \mathcal{J}. \tag{1}$$

$$\text{s.t.} \quad \bigcup_{j \in \mathcal{J}} C_j = \mathcal{M}, \quad \bigcap_{j \in \mathcal{J}} C_j = \varnothing, \tag{2}$$

$$\Big| \sum_{m \in C_j} B_m - \frac{\sum_{m \in \mathcal{M}} B_m}{J} \Big| \le \varepsilon \cdot \frac{\sum_{m \in \mathcal{M}} B_m}{J}, \forall j \in \mathcal{J}. \tag{3}$$

Here, the variable $dist(m_1, m_2)$ calculates the distance between two servers m_1 and m_2. The term $\max_{m_1, m_2 \in C_j} dist(m_1, m_2)$ in Equation (1) represents the server cluster diameter, which is defined as the maximum distance between servers within a server cluster j. In Equation (3), the hyper-parameter ε tunes the partition imbalance. The objective (1) ensures that the server cluster comprises adjacent servers, while constraint (2) ensures that server clusters do not overlap. Additionally, constraint (3) guarantees balanced resource distribution among server clusters.

We define our requests allocation solution as a set of solutions, where each solution is the request allocation strategy for each server pool, i.e., the problem of each server pool can be solved separately, and then the solutions of all server pools are combined to form the global bandwidth demand allocation solution. Specifically, each server pool's request allocation strategy includes a set of decision variables $D_{n,m}^{t,j}$ that determine the bandwidth demand of service type n on server m at time t in server pool j. In addition, the cost f_m of a server m depends on its billable bandwidth D'^{j}_{m} and the bandwidth unit price w, which is denoted by the following relationship [20]–[22]:

$$f_m = w * D'^{j}_{m} \tag{4}$$

where the bandwidth unit price w is a fixed constant. Hence, the formulation of the requests allocation problem can be stated as follows.

$$\text{Minimize} \quad \sum_{m \in C_j} D'^{j}_{m}, \forall j \in \mathcal{J}. \tag{5}$$

$$\text{s.t.} \quad \sum_{m \in C_j} D_{n,m}^{t,j} = D_n^{t,j}, \forall t \in \mathcal{T}, n \in \mathcal{N}, \tag{6}$$

$$\sum_{n=1}^{N} D_{n,m}^{t,j} \le B_m, \forall t \in \mathcal{T}, m \in \mathcal{M}, \tag{7}$$

$$D_{n,m}^{t,j} = 0, \forall t \in \mathcal{T}, m \in \mathcal{M}, n \in \mathcal{N}, A_m^n = 0. \tag{8}$$

Here, D'^{j}_{m} represents the billed bandwidth of server m under burstable billing. The calculation method is defined as follows.

$$\begin{aligned} D'^{j}_{m} &= \text{sort}(\mathcal{D}_m^j)[\alpha \times T], \\ \mathcal{D}_m^j &= \{D_m^{t,j}\}_{t=1}^{T}, \\ D_m^{t,j} &= \sum_{n \in \mathcal{N}} D_{n,m}^{t,j}, t \in \mathcal{T}, \end{aligned} \tag{9}$$

Here, $\mathcal{D}_m^j$ is a sequence of bandwidth usage records for servers m, and the function sort$(\cdot)$ sorts the sequence in ascending order. The term $[\alpha \times T]$ denotes the $\alpha \times T$-th value of the sequence, where α is a parameter for burstable billing (e.g., if using the 95th percentile billing, $\alpha = 0.95$).

Equation (6) ensures that requests are allocated during each time interval, Equation (7) prevents a server from accepting requests that exceed its available bandwidth resources, and Equation (8) ensures that requests for specific services are only allocated to servers available to provide them.

E. Complexity Analysis: Proof of NP-Completeness

To prove the NP-completeness of our edge cloud resource allocation problem, we reduce it to the set partitioning problem, which is known to be NP-complete.

Consider an instance with a single service, M servers, 95th percentile billing, and $T = 20$ time slots. Define the request sequence $\mathcal{D}$ as:

$$\mathcal{D} = \left\{ 0, 0, \ldots, 0, \frac{1}{2}\sum_{m=1}^{M} B_m, \frac{1}{2}\sum_{m=1}^{M} B_m \right\} \tag{10}$$

where B_m denotes the bandwidth of server m. The bandwidth demands for the last two time slots are half of the total bandwidth across all servers. Due to the 95th percentile billing and 20 total time slots, each server has one free time slot. In the optimal case, each server is used only once, resulting in a total bandwidth cost of 0. To achieve this, we need to partition the servers into two groups with equal total bandwidth, to handle the traffic for the last two time slots.

Since the set partitioning problem is a partition of a set of numbers into two subsets such that the sum of each of these two subsets is the same, the problem can be mapped to the set partitioning problem: Given a set $S$of bandwidth requirements, partition Sinto two subsets $a$and Bsuch that:

$$\sum_{s \in A} s = \sum_{s \in B} s \tag{11}$$

Since the set partitioning problem is NP-complete [23], and our problem can be reduced to it, our problem is also NP-complete. [24] Hence, finding an optimal solution for our problem is computationally infeasible in polynomial time.

III. Geography-Aware Server Cluster Partitioning

In this section, we analyze the necessity of cluster partitioning, highlighting its importance in optimizing resource allocation and management. Furthermore, we propose a geography-aware server cluster partitioning algorithm specifically designed for large-scale edge-cloud systems, which effectively addresses the challenges posed by geographic dispersion and spatial heterogeneity.

A. Necessity Analysis of Cluster Partitioning

In the realm of edge cloud systems, where server nodes proliferate, the unique challenge lies in the sheer abundance of servers. Reports indicate that these systems often boast thousands of server nodes [25]. With the emergence of fog computing, the inclusion of numerous embedded devices as computing nodes is a plausible scenario, potentially further amplifying the node count.

The surplus of edge cloud nodes poses a formidable challenge for real-time scheduling algorithms. Each node serves as the linchpin for handling requests, and the escalating node count inevitably expands the solution space, complicating the efficient resolution of scheduling problems. Therefore, we need a strategy to overcome this complexity, namely cluster partitioning. We posit that the computational load of a scheduling algorithm, denoted as $U(M)$, is intrinsically linked to the node count M. If the servers are partitioned into Q subsets for solving the problem, we define $\mathcal{Q} = \{1, 2, 3, \ldots, Q\}$, where $q \in \mathcal{Q}$ represents one of the subsets in the set of all subsets $\mathcal{Q}$. Since the computational load for a subset with $\frac{M}{Q}$ servers is represented as $U\left(\frac{M}{Q}\right)$, the overall workload can be expressed as $\sum_{q=1}^{Q} U\left(\frac{M}{Q}\right)$.

The premise is that as long as the time complexity of our algorithm is greater than or equal to $O(M)$, the following relationship holds:

$$\sum_{q=1}^{Q} U\left(\frac{M}{Q}\right) \leq U(M). \tag{12}$$

Essentially, the higher the algorithmic complexity, the more pronounced the reduction in computational load. Importantly, we contend that expecting a scheduling algorithm with a time complexity lower than $O(M)$ is improbable, as scheduling inherently demands comprehensive information on the bandwidth resources of all servers.

Post-cluster partitioning, executing traffic scheduling optimization may appear counter-intuitive as optimal solutions obtained within each cluster may not seamlessly combine into a globally optimal solution. The bandwidth optimization problem lacks the optimal substructure property [26]. Nevertheless, post-partitioning request scheduling aligns with the nature of proximity computing in edge clouds. To maintain quality of service, requests are strategically confined to servers within close geographical proximity, avoiding dispatch across excessively distant regions. In subsequent sections, we delve into empirical demonstrations of the impact of bandwidth cluster partitioning on cost optimization—focusing on the delicate balance between service quality and cost efficiency.

B. Cluster Partitioning Method

In this part, we delve into the nuances of our cluster partitioning method, meticulously designed to accomplish two paramount objectives:

- Ensuring High Availability: Our partitioning method is crafted to maintain equilibrium among server clusters concerning bandwidth resources. This equilibrium fortifies system robustness and minimizes the probability of user requests failing to locate available servers. Additionally, it sets the stage for bandwidth cost optimization algorithms, particularly crucial under resource constraints.

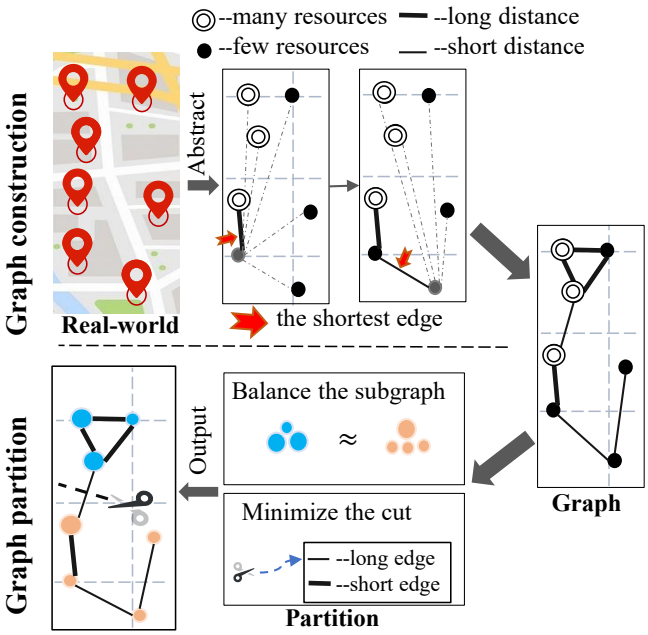

Fig. 3. The graph construction and partition.

- Guaranteeing Quality of Service: The core ambition of edge cloud clusters is to diminish service latency. To achieve this, we strive to ensure that servers within a cluster are in close proximity. This strategic proximity minimizes the occurrence of requests being scheduled to servers at considerable distances within the same cluster, thereby enhancing overall user experience.

To realize these goals, we frame server cluster partitioning as a balanced graph partitioning problem. As shown in Fig. 3, we solve the server clusters partition problem by two steps, including graph construction and graph partitioning. Graph construction and graph partitioning are sequential processes, with the output of the former serving as the input for the latter. Since the dataset collected from servers is not directly partitionable, we first construct a graph that includes node sets, node attributes, edge sets, and edge attributes. This constructed graph is then processed through graph partitioning. The approach involves partitioning the node set V into subsets while severing inter-subset edges. Simultaneously, we maintain balance within each subset, ensuring that the sum of weights, which represents bandwidth resources, is similar, i.e., adhering to the constraints defined in (3). Furthermore, we aim to minimize the sum of weights associated with all cut edges:

$$\text{Minimize} \quad \sum_{i<j} W_{i,j} \quad i \in V_q, j \in V_p, p \neq q. \tag{13}$$

Prior to engaging in graph partitioning, the essential step of graph construction is undertaken to facilitate the subsequent graph partitioning algorithm. Its primary purpose is to ensure an equitable distribution of bandwidth resources within each server cluster while minimizing the diameter of each cluster. The following sections detail the graph construction and partitioning algorithms, highlighting their key roles in achieving the objectives.

Algorithm 1: Graph Construction Algorithm

```
Initialize an empty edge set E.
for m = 1 to M do
    Initialize the set S[m] of nodes closest to distance m.
    Search for the k nearest nodes to distance m.
    Store the k nearest nodes in S[m].
end for
for m = 1 to M do
    for i in S[m] do
        if e_i^m not in E then
            Embed e_i^m to E.
        end if
    end for
end for
```

1) Graph Construction Algorithm

As illustrated in Algorithm 1, the process begins with the initialization of the edge set E and the node set V. The edge connecting node m to node i is denoted by e_i^m, where each node represents a server with associated two-dimensional coordinates that reflect its geographical location in the real world. Specifically, for each server m, the algorithm identifies k closest servers that have not yet been connected. By calculating the geographical proximity, the algorithm strategically selects these k nearest servers ($k = 10$ is used in this paper) and establishes edges between server m and the chosen nodes. This approach ensures a more balanced and efficient graph construction by leveraging spatial information to guide the connection process.

2) Graph Partitioning Algorithm

Our choice of parallel graph partitioning algorithm for complex networks brings an intelligent and scalable approach to solving the partitioning problem. This algorithm employs a multi-level graph partitioning approach, optimizing the coarsening and refinement stages through mechanisms such as size constraints and label propagation. This strategy leverages the hierarchical structure inherent in complex networks and utilizes the highly parallelizable evolutionary algorithm *KaFFPaE* [27] for high-quality graph partitioning.

The primary steps of the algorithm include:

- Label Propagation: Initially, each node resides in its own block, and its block identification is set to its node identification. The algorithm operates in rounds, traversing nodes in a random order during each round. Upon visiting a node, it moves to the block with the strongest connection until convergence is attained.
- Cluster Contraction: The graph obtained after label propagation undergoes further clustering to create a coarser granularity. This involves merging blocks, where each block in a cluster is merged into a single node. The weight of the node is set to the sum of weights of all nodes in the original block. This process ensures that the coarse

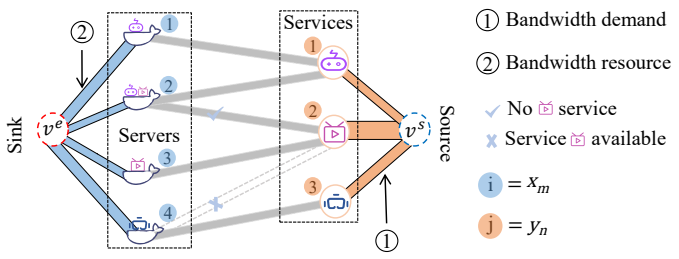

Fig. 4. An example for constructing the topology network.

graph's partition corresponds to a partition of the fine-grained graph with the same cut and balance.

- Evolutionary Algorithm Partitioning: Leveraging the *KaFFPaE* algorithm, we conduct coarse-grained evolutionary partitioning. Each processor has its own partition and a copy of the graph. After creating local partitions, each processor performs combination and mutation operations on its local partition. The algorithm guarantees that edges cut in the input partition are not contracted, preserving the superior quality of the better input individual. Moreover, *KaFFPaE* incorporates a scalable communication protocol akin to random rumor spreading for population mixing.

This graph partitioning algorithm not only demonstrates robust engineering implementation but also emphasizes the thoughtful consideration of parallelization schemes, making it well-suited for the challenges of large-scale networks—an attribute particularly beneficial for the myriad of server nodes in edge cloud clusters.

IV. Context-Capacity-Aware Task Allocation

In this section, we transform the task allocation problem into a maximum flow problem to simplify resource distribution. Additionally, we propose a context-capacity-aware task allocation algorithm that considers node capacities and contextual information to better match task demands with resources in dynamic environments.

A. Problem Transformation

After completing the server cluster partitioning, our attention turns to the request allocation algorithm, a critical phase in optimizing costs within a specific server cluster, denoted as j.

In this phase, we embark on constructing a topology network within the server cluster. Each server m is represented as node x_m, and each service n as node y_n. The establishment of links l_n^m between node y_n and node x_m occurs for each $a_m^n = 1$. Additionally, we introduce an ending node v^e and a starting node v^s to the network. Links l_m^e from each x_m to v^e and links l_s^n from v^s to each y_n are incorporated into the network. The construction process is visually depicted in Fig. 4.

Next, we define the capacity of each link l as $c(l)$:

$$\begin{aligned} &c(l_n^m) = +\infty, \quad \forall 1 \le m \le M, 1 \le n \le N, A_n^m = 1. \\ &c(l_n^m) = 0, \quad \forall 1 \le m \le M, 1 \le n \le N, A_n^m = 0. \\ &c(l_m^e) = B_m, \quad \forall 1 \le m \le M. \\ &c(l_s^n) = D_n^{t,j}, \quad \forall 1 \le n \le N, 1 \le t \le T. \end{aligned} \tag{14}$$

Algorithm 2: Bandwidth Demand Allocation Algorithm

```
Initialize topology network shown in Fig. 4
for t = 1 to T do
    Reset the topology network
    for i = 1 to N do
        c(l_s^i) = D_i^{t,j}
    end for
    for i = 1 to M do
        Set c(l_i^e) by capacity-aware adaption mechanism
    end for
    Perform maximum flow algorithm
    for i = 1 to M do
        Set weight of l_i^e by context-aware index mechanism
    end for
    Perform minimum cost maximum flow algorithm
    for i = 1 to M do
        Store the flow of l_i^e
    end for
end for
Obtain D_m^j based on Ar_m
Calculate D'^j_m based on Equation (9)
```

This construction results in a topology network featuring a single-source v^s and a single-sink v^e, as illustrated in Fig. 4. To obtain a solution that adheres to the constraints specified in (6), (7), and (8), we transform the bandwidth demand allocation problem into a ***max flow problem***. Seeking the max flow from the source to the sink provides a bandwidth demand allocation solution in line with all defined constraints.

Within this network, the flow from node x_m to node v^e symbolizes the total bandwidth demand handled by server m. The flow is confined by the capacity of link l_m^e, thereby restricting the load on server m.

Additionally, the flow from node y_n to node x_m represents the bandwidth demand of service n allocated to bandwidth m. This allocation is permissible when the server is available and forbidden otherwise.

Moreover, the flow from node v^s to y_n denotes the bandwidth demand of service n that has been allocated. This flow is subject to the constraint of link capacity $c(l_s^n)$ under normal circumstances. Thus, we can effectively transform the bandwidth demand allocation problem into a max flow problem by this transformation.

B. Algorithm Design

Subsequently, we fully leverage the characteristics of burstable billing. As introduced in Sec. I-A, 95th percentile billing determines the bandwidth cost based on the 95th percentile of bandwidth usage. Therefore, the highest 5% of the sampled bandwidth values, whatever their value, do not affect the bandwidth cost and should be utilised as much as possible. Conversely, the lowest 95% of sampled bandwidth values cannot reduce the bandwidth cost, no matter how low they are, hence they need to be kept below and as close to the 95th percentile

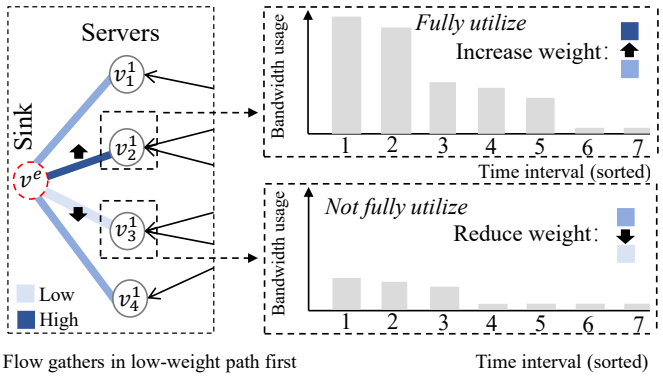

Fig. 5. A context-aware tune for the link weight.

usage as possible. Based on this principle, we introduce an enhanced burstable billing optimization method employing two mechanisms: *context-aware index* and *capacity-aware adaptation*. The detailed procedure is outlined in Algorithm 2.

1) *Context-Aware Index Mechanism*

In this mechanism, we introduce an approach to adjust the weights assigned to individual servers based on their historical bandwidth usage. The objective is to intelligently guide the flow of bandwidth demands, optimizing the allocation process using the minimum cost flows algorithm.

The bandwidth usage of server m is characterized by $sort(\mathcal{D}_m^j)[\alpha \times T]$ when employing burstable billing. Consequently, this measurement remains unaffected by the extreme bandwidth consumed in the initial $\alpha \times T - 1$ time intervals, referred to as "free time intervals". Leveraging these free time intervals is pivotal for cost minimization.

To evaluate the optimal utilization of these free time intervals for server m, we introduce the context-aware index $\mathbb{I}_m$. This index is dynamically determined by maintaining a sorted array Ar_m for each server, recording bandwidth usage in each time interval. The length of Ar_m is T, with each element representing bandwidth usage in a time interval. Continuous updates to Ar_m require constant sorting, and $\mathbb{I}_m$ is defined as $Ar_m[\alpha \times T - 1]$, varying with the context message Ar_m.

Fig. 5 illustrates this dynamic adjustment, where the weight of link l_m^e adapts based on $\mathbb{I}^{max} - \mathbb{I}_m$, with $\mathbb{I}^{max}$ as a hyper-parameter exceeding $\mathbb{I}_m$ for all $1 \leq m \leq M$. This context-aware tuning ensures adaptive link weights that are consistent with the dynamics of bandwidth and optimizes the flow distribution.

The mechanism utilizes temporal patterns of bandwidth usage to improve resource allocation efficiency. Its adaptability to different server conditions makes it a valuable tool to promote cost-effective and intelligent task allocation strategies.

2) *Capacity-Aware Adaption Mechanism*

This mechanism capitalizes on the untapped capacity of servers, orchestrating a strategic bandwidth allocation approach that optimizes efficiency and cost-effectiveness. Its core strategy involves the dynamic adjustment of link capacities based on paid capacity information, followed by a thoughtful distribution of flows to maximize the utilization of available free capacity.

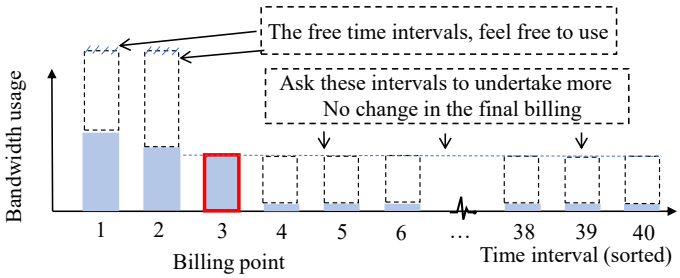

Fig. 6. An illustration for free intervals in 95th percentile billing.

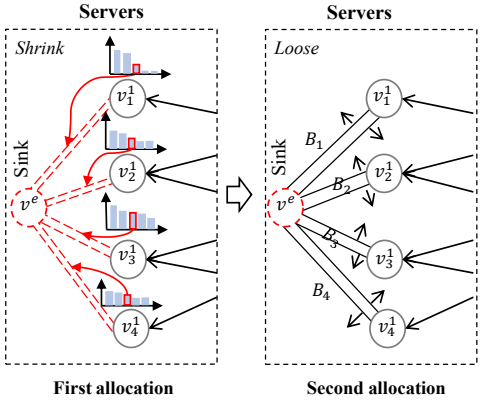

Fig. 7. An example for twice allocation in capacity exploitation.

In Fig. 6, the temporal dynamics behind the burstable billing measure point unveil intervals of untapped bandwidth capacity. Referred to as "free capacities", these intervals present an opportunity for further cost reduction.

The mechanism employs a two-step process to harness this potential. Initially, it adjusts the capacity of link l_m^e using the bandwidth usage array Ar_m. Setting the capacity to $Ar_m[\alpha \times T]$, the max flow algorithm is applied for the initial allocation. Subsequently, the capacity of l_m^e is readjusted to B_m, and the max flow algorithm is reapplied to obtain the final flow allocation. The cumulative effect of these allocations results in the ultimate allocation, as depicted in Fig. 7.

By restricting the link capacity in the initial allocation, the bandwidth consumption of server m is strategically capped below the measure point $Ar[\alpha \times T]$. This deliberate approach in the first allocation aims to maximize the utilization of available free capacity. The subsequent application of context-aware mechanisms during the second allocation process is greatly eased by the judicious use of free capacity in the initial step. This capacity-aware adaptation can effectively utilize the capabilities of the server while minimizing the cost. It demonstrates the potential of the adaptation mechanism to improve the efficiency of bandwidth allocation in dynamic computing environments.

V. Performance Evaluation

In this section, we first describe the experimental setup, including the trace, baseline algorithms, and hyper-parameter settings. We then perform a comprehensive performance analysis and validation of cluster partitioning and task allocation, evaluating the effectiveness of methods.

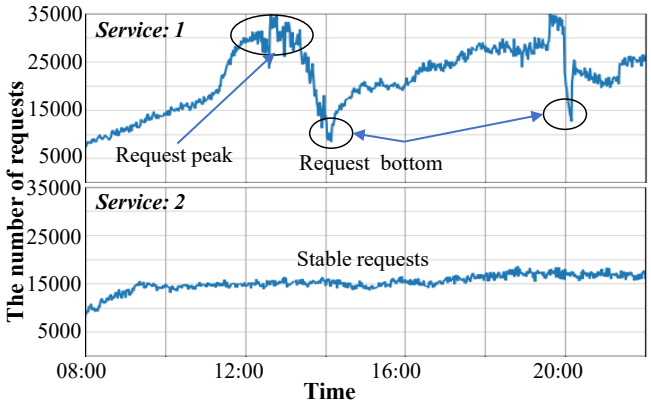

Fig. 8. Request data distribution of two services.

A. Evaluation Setup

To ensure experiment credibility, we first introduce the evaluation setup. The following describes in detail the settings used during the evaluation process, providing a comprehensive understanding of the conditions for performance evaluation.

1) Real-world Request Traces

To assess the effectiveness of our framework, we employ real-world request traces from an edge-cloud operator[4]. These traces, spanning over 10 million user requests from October 1, 2021, to October 29, 2021, provide a comprehensive dataset of approximately 418 GB. Fig. 8 illustrates the distribution of request data for two services over a single day, offering insights into temporal dynamics and service demands. Additionally, data on the geographical location and bandwidth capacity of around 1,500 servers complement the evaluation. Since only the latitude and longitude of the server can be obtained in the data set, we use the geographical location in the process of graph partitioning, but the proposed algorithm is a general scheme, and it is also feasible for network hops or communication delays, etc., as a measure of distance.

2) Baseline Algorithms

The baseline algorithms utilized in our experiments are categorized into two key aspects. First, to validate the ***performance of cluster partitioning***, we employed the following methods:

- ***Average-random***: Divide the servers into multiple server clusters as uniformly random as possible;
- ***K-Means*** **[28]**: The servers are partitioned iteratively based on the center distance.
- ***Graph Partition based on Backtracking (GPB)*** **[29]**: Construct a spanning tree based on the adjacency list, traverse all root-to-leaf paths to identify independent sets for graph partitioning.

Second, to evaluate the ***performance of task allocation***, we utilized the following baseline methods:

- ***Cascara*** **[18]**: It optimizes bandwidth allocations with nonlinear pricing by using latency-equivalent peer links on the cloud edge;

[4]PPIO Edge Cloud, PPIO Cloud Computing (Shanghai) Co., Ltd., https://www.ppio.cn

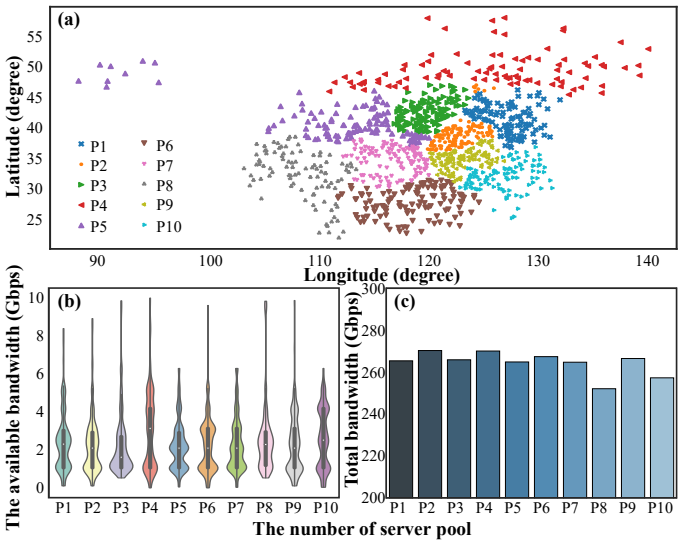

Fig. 9. The distribution of server clusters (a) and the contrast of the available bandwidth (b) and the total bandwidth (c).

- ***GeoCo***: It only uses the context-aware index mechanism to optimize bandwidth allocation;
- ***GeoCa***: It only uses the capacity-aware adaption mechanism to optimize bandwidth allocation;
- ***Greedy*** **[30]**: It prioritizes servers with more available bandwidth when making task allocations;
- ***SM-CTP*** **[31]**: Iteratively select task sets that maximize the submodular function to derive the optimal task allocations.

3) Hyper-parameter Settings

For the proposed algorithms, we configure the hyper-parameters with $\epsilon = 0.1$ and $\beta = 1$ for the server cluster partitioning, while setting $\alpha = 0.95$ for bandwidth demand allocation. These parameters are chosen to balance precision and efficiency, ensuring robust performance across various network environments and system constraints. They allow the algorithms to effectively manage trade-offs between accuracy and speed for diverse real-world applications. Additionally, the system cost is calculated based on Equation (4), with w set to 1.5.

B. Evaluation Results

1) Performance of Cluster Partitioning

We first validate the cluster partitioning algorithm to ensure that multiple homogeneous server clusters can be partitioned based on heterogeneous servers. In this way, on the one hand, the impact of heterogeneity on subsequent task allocation can be reduced, and on the other hand, the negative impact of the large number of servers on the algorithm can be reduced.

First, as shown in Fig. 9 (a), *Geo* effectively divides the geographically dispersed servers into 10 subsets and the servers in each subset are geographically centralized. In terms of server resources, *Geo* can keep the available bandwidth resources and the total bandwidth of each server cluster relatively balanced despite the strong resource heterogeneity of these servers (as shown in Fig. 9 (b) (c)).

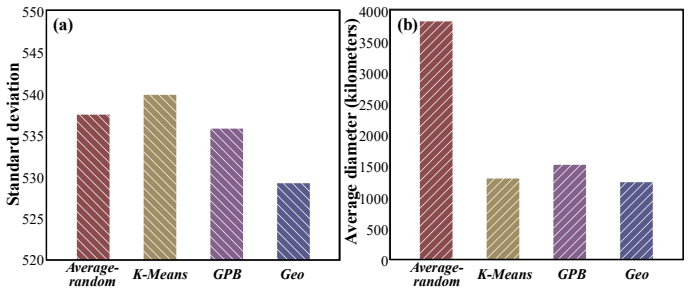

Fig. 10. Comparison of standard deviation of bandwidth resources (a) and average diameter (b) of server clusters for different algorithms.

As shown in Fig. 10 (a), *Geo* has the smallest standard deviation, which can maximally guarantee that different server clusters have the same resource scale. On this basis, as shown in Fig. 10 (b), the servers in the server cluster divided by *Geo* are the most concentrated, and less transmission distance is beneficial to further reduce the transmission delay during task allocation.

2) Performance of Task Allocation

As shown in Figure 11, we conducted further comparisons of the system cost (defined as f_m in Equation (4) with $w = 1.5$), thereby verifying the performance under different application scenarios.

Figure 11 (a) illustrates the system cost concerning the average number of services across 130 servers. The proposed *GeoCo* and *GeoCC* algorithms outperform the greedy anytime algorithm. However, the degree of performance improvement is less pronounced with a smaller average number of services compared to a larger one. This variation is attributed to the increased constraints on the allocation process when the average service number is small, leading to a reduction in flexibility.

Figure 11 (b) depicts the system cost in relation to the number of servers. An expansion in the number of servers results in more available free time intervals, contributing to a noteworthy reduction in the overall system cost, as illustrated in the figure. This observation confirms the efficacy of the context-aware and capacity-aware mechanisms in capitalizing on the free time intervals facilitated by burstable billing. Conversely, insufficient servers impede bandwidth demand allocation, emphasizing the importance of a well-balanced server cluster partition.

Figure 11 (c) illustrates the system cost concerning user bandwidth demands. We simulate variations in demands by scaling bandwidth requests with different factors. As the demands escalate, a corresponding rise in the cost is observed. This outcome is associated with the burstable billing feature, which accommodates a specific threshold of burst bandwidth demands. Uncontrolled increases in demand inevitably lead to escalated costs for the system.

Figure 11 (d) depicts the influence of the server cluster partition number on server cluster diameter and its impact on bandwidth demand allocation. We conduct experiments with various partition numbers using a real-world server set, keeping the services or demand sum unchanged. On one hand, we note that the cost rises with an increasing partition number due to heightened constraints on allocation. On the other hand, a smaller partition number leads to a larger server cluster diameter, reducing the locality of bandwidth demand allocation and prolonging the execution time of the allocation algorithm.

Examining all subfigures in Fig. 11, it is evident that the greedy algorithm exhibits the poorest performance. This can be attributed to the intricate nature of burstable billing, where some servers are prone to being overused. In most cases, *Cascara* and *SM-CTP* perform worse than *GeoCC*. *Cascara* neglects the intricate matching relationship between services and demands in the edge-cloud system, resulting in suboptimal performance despite the utilization of prior information. *SM-CTP* lacks fine-grained awareness of network connectivity and cannot fully utilize bandwidth resources. Compared to *Cascara*, which has the closest performance, *GeoCC* can reduce bandwidth costs by an average of 28.30%, with a maximum reduction of 81.83%.

VI. Related Work

In this section, we review the research progress in the relevant field and highlight the innovations and differences of our study compared to existing work.

A. Task Allocation in the Edge Cloud

In previous research work, there are two main goals of task allocation in edge cloud or even traditional cloud cluster. The first is to improve the user service experience, such as reducing the packet loss rate of transmission and reducing the delay of real-time streaming service. The second is to reduce the cost of services, such as reducing bandwidth overhead, improving machine utilization, reducing energy consumption, etc.

Oo et al. [32] proposed a perceptual allocation algorithm that takes into account the varying tolerance to latency for different types of tasks. The decision on whether to forward a task to the central cloud or to perform computations at the edge is based on the task's specific latency requirements. They demonstrated that this allocation strategy is more effective in meeting the computational task demands compared with the allocation algorithms that do not consider the diverse latency requirements of computational tasks. Li et al. [33] introduced an energy-aware edge service placement algorithm with the aim of reducing the overall energy consumption costs of servers while maintaining acceptable user access latency. This work models the problem as a multi-objective optimization problem and employs a particle swarm optimization-based energy-aware edge service placement algorithm to solve it. He et al. [34] proposed an allocation method to enhance offloading efficiency and reduce application latency in resource-constrained edge devices. The algorithm, using a genetic algorithm, encodes tasks generated by edge devices into binary code to optimize task allocation and reduce task latency. The allocation strategy considers the dependency between computational tasks. This work models the allocation problem as a joint task assignment and flow scheduling problem, utilizing a data flow programming model to analyze the correlations between tasks and thereby improve task throughput.

The above research works optimize task allocation in edge cloud with different objectives, and their optimization objec-

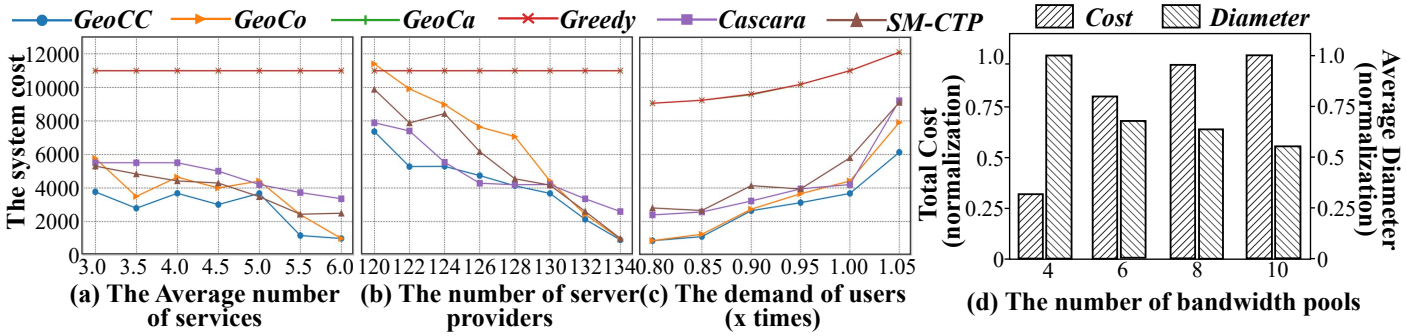

Fig. 11. Cost comparison of the algorithms under different conditions (a) (b) (c) and the impact of partition number on cost and diameter (d).

tives are delay, energy consumption, task throughput, etc. This reflects that there are often multiple objectives to be considered in the optimization of task allocation in edge cloud. However, none of the above works consider the burstable billing mode and cannot maximize the resource cost reduction.

B. Task Allocation to Optimize Costs

Task allocation can reduce the bandwidth cost as much as possible under the condition of satisfying service constraints, which has application potentials and has attracted research attentions. There are generally two main ideas for cost optimization schemes. The first is to dynamically select low-cost links for data transmission to save costs. The second is to reduce the amount of data transmission through caching and other means to reduce the cost.

García-Dorado et al. [35] proposed a cloud application system architecture that comprehensively considers the transmission cost and data redundancy. By constructing a batch data transfer overlay distribution tree, they optimize the distribution cost while ensuring that end-to-end data transmission remains unaffected. This work monitors the throughput between TCP data centers and suggests alternative tree structures that respect the original transmission time, thereby saving infrastructure costs in cloud computing centers. Ahmed et al. [36] maintain multiple transmission paths from content distribution servers to users. Due to the dynamic nature of performance and pricing along transportation routes, a routing selection strategy is implemented to optimize the trade-off between performance and cost. This work formalizes the routing problem in a multi-attribute manner while simultaneously optimizing performance and cost. Le et al. [13] use an auction framework to simulate interactions between network providers and tenants in the bandwidth market. They propose a solution based on a stochastic auction mechanism that determines the bandwidth quantity and corresponding payment based on bids submitted by tenants, aiming to maximize social welfare.

While the aforementioned studies optimized server bandwidth costs, they did not specifically target the optimization of burstable billing mechanisms. Burstable billing mechanisms are widely adopted by network service providers, introducing unprecedented complexity to optimization problems. Therefore, there is a need for targeted optimization solutions to address the challenges posed by burstable billing.

C. Task Allocation for Burstable Billing

The burstable billing mechanism is a widely adopted billing method by internet service providers [37]. Its main characteristic is the tolerance of brief periods of high bandwidth usage, resulting in relatively higher charges for applications with significant demand fluctuations. Typically, operators employ the 95th percentile billing to calculate bandwidth costs [10]. This method allows for 5% of the highest bandwidth usage moments within a billing period, with the billing bandwidth determined by the 95th highest usage level during the entire billing cycle.

Lin et al. [38] proposed a cost-optimized video streaming scheduling system that delays the delivery of video streams to prevent an increase in current burstable billing costs. Li et al. [39] introduced a mechanism based on cumulative traffic, utilizing Lyapunov optimization techniques to design a pricing-aware control framework. This framework schedules more traffic peaks during free slots while maintaining lower traffic differentials in the remaining slots. However, both of these approaches are not suitable for tasks with high demand and strong real-time constraints. Postponing the delivery of tasks is unacceptable for many tasks in the new generation of the internet, limiting the applicability of these methods.

Singh et al. [40] computed cost values under burstable billing in advance using historical prior information. Subsequently, available costs were allocated based on the size of server bandwidth resources, and a greedy approach was employed to minimize costs while utilizing the allocated resources as much as possible without significantly affecting latency. However, this work heavily relies on prior information from historical data center data, making the resulting algorithm generalize. Additionally, the centralized setting of data centers in this work is relatively straightforward and does not consider the diversity of services and the availability of servers tailored to specific services.

VII. Conclusion

In this paper, we propose a geography-context-capacity aware bandwidth cost economizing strategy in the edge-cloud system, i.e., *GeoCC*. *GeoCC* adopts server cluster partition and the optimal burstable billing mechanisms. Here, a geography-aware graph construction and partition algorithm is designed. Meanwhile, an improved burstable billing optimization mechanism is also proposed. Finally, the evaluation using real-world tracking

from an edge cloud company verified that *GeoCC* can effectively reduce bandwidth peaks according to the billing model, resulting in a significant reduction in the cost of bandwidth in burstable billing.

REFERENCES

[1] Y. Li, C. Qiu, X. Wang, C. Zhang, W. Wang, S. Lan, and J. Jiang, "Lococa: Location-context-capacity aware cost economizing in edge-cloud systems," in *Global Communications Conference*. IEEE, 2023.

[2] Y. Li, Z. Han, Q. Zhang, Z. Li, and H. Tan, "Automating cloud deployment for deep learning inference of real-time online services," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1668–1677.

[3] P. Jiang, C.-K. Wen, S. Jin, and G. Y. Li, "Wireless semantic communications for video conferencing," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 1, pp. 230–244, 2022.

[4] S. T. Ahmed, V. Kumar, and J. Kim, "Aitel: ehealth augmented intelligence based telemedicine resource recommendation framework for iot devices in smart cities," *IEEE Internet of Things Journal*, 2023.

[5] M. Xu, Z. Fu, X. Ma, L. Zhang, Y. Li, F. Qian, S. Wang, K. Li, J. Yang, and X. Liu, "From cloud to edge: a first look at public edge platforms," in *ACM Internet Measurement Conference*, 2021, pp. 37–53.

[6] S. Shen, Y. Feng, M. Xu, C. Zhang, X. Wang, W. Wang, and V. C. Leung, "A holistic qos view of crowdsourced edge cloud platform," in *International Symposium on Quality of Service*. IEEE/ACM, 2023, pp. 01–10.

[7] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial internet of things: Architecture, advances and challenges," *Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020.

[8] 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," *Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1160–1192, 2021.

[9] Q. Hu, Q. Peng, J. Shang, Y. Li, and J. He, "Eba: An adaptive large neighborhood search-based approach for edge bandwidth allocation," in *International Conference on Collaborative Computing: Networking, Applications and Worksharing*. Springer, 2022, pp. 249–268.

[10] Y. Zhan, M. Ghamkhari, H. Akhavan-Hejazi, D. Xu, and H. Mohsenian-Rad, "Cost-aware traffic management under demand uncertainty from a colocation data center user's perspective," *IEEE Transactions on Services Computing*, vol. 14, no. 2, pp. 400–412, 2018.

[11] J. Maghakian, R. Lee, M. Hajiesmaili, J. Li, R. Sitaraman, and Z. Liu, "Applied online algorithms with heterogeneous predictors," in *International Conference on Machine Learning*. PMLR, 2023, pp. 23 484–23 497.

[12] C. Bhar and E. Agrell, "Energy- and bandwidth-efficient, QoS-aware edge caching in fog-enhanced radio access networks," *IEEE Journal on Selected Areas in Commu.*, vol. 39, no. 9, pp. 2762–2771, 2021.

[13] T. H. T. Le, N. H. Tran, T. LeAnh, T. Z. Oo, K. Kim, S. Ren, and C. S. Hong, "Auction mechanism for dynamic bandwidth allocation in multi-tenant edge computing," *IEEE Trans. on Veh. Tech.*, vol. 69, no. 12, pp. 15 162–15 176, 2020.

[14] Y. Sun, Z. Chen, M. Tao, and H. Liu, "Bandwidth gain from mobile edge computing and caching in wireless multicast systems," *IEEE Trans. on Wireless Commu.*, vol. 19, no. 6, pp. 3992–4007, 2020.

[15] J. L. García-Dorado and S. G. Rao, "Cost-aware multi data-center bulk transfers in the cloud from a customer-side perspective," *IEEE Trans. on Cloud Comp.*, vol. 7, no. 1, pp. 34–47, 2019.

[16] W. Li, X. Yuan, K. Li, H. Qi, and X. Zhou, "Leveraging endpoint flexibility when scheduling coflows across geo-distributed datacenters," in *INFOCOM*, 2018, pp. 873–881.

[17] X. Guo, R. Singh, T. Zhao, and Z. Niu, "An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems," in *ICC*, 2016, pp. 1–7.

[18] R. Singh, S. Agarwal, M. Calder, and V. Bahl, "Cost-effective cloud edge traffic engineering with cascara," in *USENIX NSDI*, 2021.

[19] T. Q. Dinh, B. Liang, T. Q. S. Quek, and H. Shin, "Online resource procurement and allocation in a hybrid edge-cloud computing system," *IEEE Trans. on Wireless Commu.*, vol. 19, no. 3, pp. 2137–2149, 2020.

[20] G. labs, "Understand your grafana cloud metrics invoice," https://grafana.com/docs/grafana-cloud/cost-management-and-billing/understand-your-invoice/metrics-invoice/#95th-percentile-billing.

[21] A. cloud, "Monthly 95th percentile bandwidth of alibaba cloud," https://www.alibabacloud.com/help/en/ens/product-overview/billing-for-bandwidth#6ad7cf2cddj3o.

[22] F. cloud, "Cost and billing of fortiweb cloud," https://docs.fortinet.com/document/fortiweb-cloud/latest/user-guide/555164/cost-and-billing.

[23] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, ser. Mathematical Sciences Series. Freeman, 1979. [Online]. Available: https://books.google.co.jp/books?id=fjxGAQAAIAAJ

[24] M. R. Garey and D. S. Johnson, ""strong"np-completeness results: Motivation, examples, and implications," *Journal of the ACM (JACM)*, vol. 25, no. 3, pp. 499–508, 1978.

[25] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 160–168, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731518304398

[26] D. Thierens and P. A. Bosman, "Optimal mixing evolutionary algorithms," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011, pp. 617–624.

[27] P. Sanders and C. Schulz, "Distributed evolutionary graph partitioning," in *ALENEX*. SIAM, 2012, pp. 16–29.

[28] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[29] Z. Zhao, H. Cheng, X. Xu, and Y. Pan, "Graph partition and multiple choice-ucb based algorithms for edge server placement in mec environment," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 4050–4061, 2023.

[30] Y. Zhang, C. Chen, L. Liu, D. Lan, H. Jiang, and S. Wan, "Aerial edge computing on orbit: A task offloading and allocation scheme," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 1, pp. 275–285, 2022.

[31] L. Luo, G. Zhao, H. Xu, Z. Yu, and L. Xie, "Achieving cost optimization for tenant task placement in geo-distributed clouds," *IEEE/ACM Transactions on Networking*, 2023.

[32] T. Oo and Y.-B. Ko, "Application-aware task scheduling in heterogeneous edge cloud," in *International Conference on Information and Communication Technology Convergence*. IEEE, 2019, pp. 1316–1320.

[33] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *International conference on edge computing*. IEEE, 2018, pp. 66–73.

[34] J. He, "Optimization of edge delay sensitive task scheduling based on genetic algorithm," in *International Conference on Algorithms, Data Mining, and Information Technology*. IEEE, 2022, pp. 155–159.

[35] J. L. García-Dorado and S. G. Rao, "Cost-aware multi data-center bulk transfers in the cloud from a customer-side perspective," *Transactions on Cloud Computing*, vol. 7, no. 1, pp. 34–47, 2015.

[36] F. Ahmed, M. Z. Shafiq, A. R. Khakpour, and A. X. Liu, "Optimizing internet transit routing for content delivery networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 76–89, 2017.

[37] H. Chen, H. Zhan, H. Tan, H. Xu, W. Shan, S. Chen, and X.-Y. Li, "Online traffic allocation based on percentile charging for practical cdns," in *IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*, 2022, pp. 1–10.

[38] Y. Lin, H. Shen, and L. Chen, "Ecoflow: An economical and deadline-driven inter-datacenter video flow scheduling system," in *ACM international conference on Multimedia*, 2015, pp. 1059–1062.

[39] W. Li, X. Zhou, K. Li, H. Qi, and D. Guo, "More peak, less differentiation: Towards a pricing-aware online control framework for inter-datacenter transfers," in *International Conference on Distributed Computing Systems*. IEEE, 2017, pp. 2105–2110.

[40] R. Singh, S. Agarwal, M. Calder, and P. Bahl, "Cost-effective cloud edge traffic engineering with cascara," in *USENIX Symposium on Networked Systems Design and Implementation*, 2021, pp. 201–216.