An Online Orchestration Mechanism for General-Purpose Edge Computing

Xun Shao Member, IEEE, Go Hasegawa Member, IEEE, Mianxiong Dong Senior Member, IEEE, Zhi Liu Senior Member, IEEE, Hiroshi Masui NonMember, IEEE, and Yusheng Ji Fellow, IEEE

Abstract—in recent years, the fast development of mobile communications and cloud systems has substantially promoted edge computing. By pushing server resources to the edge, mobile service providers can deliver their content and services with enhanced performance, and mobile-network carriers can alleviate congestion in the core networks. Although edge computing has been attracting much interest, most current research is application-specific, and analysis is lacking from a business perspective. Edge cloud providers (ECPs) that provide general-purpose edge cloud services to mobile service providers and users. In this article, we present a vision of general-purpose edge computing realized by multiple interconnected edge clouds, analyzing the business model from the viewpoint of ECPs and identifying the main issues to address to maximize benefits for ECPs. Specifically, we formalize the long-term revenue of ECPs as a function of server-resource allocation and public-data-placement decisions subject to the amount of physical resources and inter-cloud data-transportation cost constraints. To optimize the long-term objective, we propose an online framework that integrates the drift-plus-penalty and primal-dual methods. With theoretical analysis and simulations, we show that the proposed method approximates the optimal solution in a challenging environment without having future knowledge of the system.

Index Terms—General-Purpose Edge Computing, Online Mechanism, Drift-Plus-Penalty Optimization, Primal-Dual Optimization

1 INTRODUCTION

1.1 Motivation

In recent years, the fast development of mobile devices, communications, and cloud computing has produced a surge of Internet-of-things services and applications. Because most mobile services and applications adopt the device-cloud architecture, mobile traffic across the Internet is dramatically increasing, becoming a challenging issue for mobile-network carriers. Due to the great distances between mobile devices and cloud datacenters, however, mobile users suffer from large latency and do not receive satisfactory service. To solve these issues, deploying server resources at the edge of the Internet and providing mobile services with edge servers is a promising method. With edge computing, mobile content, service providers, and mobile devices can take advantage of approximate computing and storage resources and alleviate congestion in the mobile core network.

One of the most interesting uses of edge computing is data caching. An ECP can cache popular data in edge clouds when the data go across their networks, and then it can serve data requests with the cached data rather than forward the requests to the Internet. Task offloading is also an extensively researched area with edge clouds. Mobile devices can offload computation-intensive tasks, such as video analysis, to edge servers to save battery and accelerate processing. Recently, more advanced, complicated edge services, such as edge computing-assisted online gaming, virtual reality, and augmented reality [1] [2] [3], have been studied and developed, showing the great potential of edge computing to develop emerging applications. Currently, however, most edge-computing research is service- and application-specific, and there is little discussion of general-purpose edge computing.

1.2 Our vision

In this work, rather than an application-specific edge-computing service, we consider edge-computing systems for general purposes. We analyze the business model, identify the main issues, and provide a solution. We illustrate the general-purpose edge computing in Fig. 1. ECPs operate multiple distributed edge datacenters, which are interconnected with high-speed networks. Recently, adopting C-RAN has been a major trend in mobile-network evolution. In this work, we consider the case of edge datacenters being co-located with C-RANs. With a distributed edge cloud network, an ECP provides general-purpose edge-computing services to mobile service providers and users like distributed cloud providers. Specifically, ECPs provide isolated environments with virtual machines (VMs) or containers for mobile devices associated with specific mobile services (identified by service ID) to execute computing tasks. The general-purpose edge-computing framework is supposed...
to support highly customized requests. Fig. 1 shows an example. In the request, the mobile device requires 2 VMs of type 1 to process data \( o_1 \) and 1 VM of type 2 to process data \( o_4 \). Notice that \( o_1 \) and \( o_4 \) already exist on the Internet, whereas \( o_2 \) is uploaded by the mobile device. When receiving such a request, an ECP should decide how to provide the VMs to satisfy the request. In Fig. 1’s example, the ECP assembles two type-1 VMs in C-RAN 1 and one type-2 VM in C-RAN 2. Notice that as \( o_1 \) is already placed in C-RAN 1, the type-1 VMs can fetch the data from the local cache, while because \( o_1 \) and \( o_2 \) are not placed in the C-RAN 2, the type-2 VM has to fetch data \( o_1 \) and \( o_4 \) across the link between C-RANs 1 and 2. With this process, the ECP obtains revenue by providing VMs like cloud providers.

Three kinds of business entities could be involved in this process - ECPs, edge service providers, and mobile users - that have flexible business relationships among them. For example, a mobile user could subscribe to a mobile service from an edge service provider, and the edge service provider has a contract with the ECP so that, when the ECP receives a request associated with the service, the ECP allocates the VMs to the mobile device and charges the edge service provider. Fig. 1 shows the monetary flow. Notice that the ECP and edge service provider are logical and conceptual. It is likely that in practice, the mobile-network carrier acts as an ECP because they operate the datacenters of C-RAN. The ECP could also be an edge service provider that provides original service to mobile users.

1.3 Challenges and contributions
From the viewpoint of ECPs, a complicated joint optimization problem exists regarding VM allocation and public data placement. As for the resource-allocation problem, due to the heterogeneous features of the physical resources of distributed edge datacenters, optimizing resource-allocation decisions without the statistical knowledge of future requests is challenging. To maximize long-term revenue, maintaining the availability of each resource in each edge datacenter is important. Allocating resources only considering proximity would deplete certain resources in certain edge datacenter facing specific request sequences. Besides revenue, the cost incurred by inter-cloud data transportation cannot be ignored. Placing popular data in distributed caches helps reduce inter-cloud data transportation, but resource-allocation and data-placement decisions have complicated interactions. For example, in Fig. 1 assembling the type-2 VM in C-RAN 1 can save the inter-cloud data-transportation cost of fetching \( o_1 \) and \( o_4 \); however, it would also cause a possible resource availability issue for C-RAN 1. Caching \( o_1 \) in C-RAN 2 earlier helps save the inter-cloud data-transportation cost if assembling a type-2 VM in C-RAN 2, but due to the size limit of the caching device, other data may have to be emitted. Without knowing the future request sequence, it is challenging to design an algorithm to make such online decisions. Even if the future request sequence can be predicted well, well-known online learning algorithms, such as the Markov decision process, fail to compute results in a reasonable period given a huge state space. In this article, by comprehensively analyzing the properties and interactions of resource-allocation and data-placement decisions, we present an online mechanism that can solve the joint optimization problem efficiently and needs no prior knowledge of future request sequences.

1.4 Article organization
The remainder of this article is organized as follows: Section 2 presents the problem definition with a mathematical model, and Section 3 presents the detailed solution, including the framework and sub-routines. The performance of the proposed method is analyzed theoretically in Section 4 and experimentally in Section 5. We then introduce related work in Section 6 and conclude the article in Section 7.

2 System model and problem formulation
In this section, we present our vision of general-purpose edge computing in detail, followed by the settings of the considered model. Finally, we formulate the time-average revenue-optimization problem subject to various constraints. Tab. 1 summarizes the key notations used in this article.

2.1 Settings
We assume that an ECP operates multiple edge clouds distributed but interconnected with a high-speed network.
Each edge cloud is indexed by $i$ (sometimes $j$). The latency between edge cloud $i$ and $j$ is denoted as $w_{i,j}$. Each edge cloud has multiple kinds of physical resources (e.g., CPU, memory, storage, etc.) that can be used to assemble specific types of VMs to mobile users. We also assume multiple kinds of resources and index specific resource with $r$. The amount of resource $r$ for assembling 1 type-$k$ VM is denoted as $g_{k,r}$. The ECP provides $K$ types of VM, each of which is indexed with $k$. The ECP obtains revenue at the rate $p_k$ for providing type-$k$ VMs per unit time. In addition to the resources for computing purposes, there are also low-cost caching devices with which the edge cloud can place popular data locally to reduce inter-cloud data-transportation costs and better serve mobile users. The size of data $o$ is denoted as $s_{o,i}$ and the size of the cache in edge cloud $i$ is denoted as $S_i$. The ECP operates caching as supplementary to reduce the operational cost, and it is transparent to mobile service providers and users. We index requests with $l$. A request contains the service identifier, time length, a set of VMs associated with the data list to process, and the data list to upload. The service identifier is for the ECP to identify the associated mobile service provider for charging purposes. The time length is used to specify the period of VM usage. The data requested could either be existing data on the Internet (public data) or uploaded from the mobile user that initiates the request (private data). For example, in Fig. 1 the requested data $o_1$ is public (i.e., can be found in the Internet), while $o_4$ is uploaded from a mobile device. We denote the set of data from request $l$ associated with the type-$k$ VM as $\mathcal{O}_{l,k}$. For a request $l$, there are various resource-allocation strategies to satisfy the request. Denoting $A^l$ as a specific feasible strategy, we associate a binary decision variable $x^l_{A} \in \{0, 1\}$ to it. Specifically, $x^l_A$ equals 1 if $A^l$ and 0 otherwise. For a request, there is no need to adopt more than one resource-allocation strategy to satisfy it. Formally, we have the following constraints on $x^l_{A}$:

$$\sum_{A \in A^l} x^l_{A} \leq 1, \quad x^l_{A} \in \{0, 1\} \quad \forall l, A. \quad (1)$$

### 2.2 Time scales

In this article, we adopt two kinds of time scales: a coarse-grained time slot indexed by $T$ and a fine-grained time slot indexed by $t$. The coarse-grained time slot captures long-term properties, such as time-average revenue, time-average transportation cost, and data popularity. We make decisions related to these relatively time-insensitive properties with the coarse-grained time slot. Besides the time-insensitive properties and tasks, there are also time-sensitive ones; for example, when a request arrives, the ECP should not buffer the request until the end of a certain time slot but must allocate resources immediately. For resource allocation, because the decision is subject to the amount of each kind of resource in each edge cloud, for every such constraint, we introduce a dual variable to evaluate the shadow price of the physical resource. Specifically, we construct the dual problem from the primal problem. The arrival of a new request $l$ introduces a new set of decision variables $x^l_{A}$ to the primal problem while introducing a new set of constraints to the dual problem. When a certain resource is allocated, we increase the dual variable (shadow price) associated with the resource exponentially to 1) reflect the rarity of the resource and 2) keep the ratio of the primal value and dual value increment a fixed value so that the solutions are always competitive with any sequence of request arrivals. However, when a request process finishes, the allocated resource is released, increasing the amount of available resources; correspondingly, the competitiveness changes. For this purpose, we further discretize each coarse-grained time slot into multiple fine-grained time slots. In a fine-grained time slot $t$, we only consider the arriving requests during $t$ and neglect the finished requests. At the end of $t$, we update the available resource amount by considering all released resources during $t$ and initiate a new primal and dual problem pair for $t + 1$. Correspondingly, the available resource amount is subscribed by $t$, and for each $t$, the following constraint must be satisfied:

$$\sum_{i \in I_T} \sum_{A \in A^l} N^l_{A,i,k} g_{r,k} x^l_{A} \leq c_{i,r,t} \quad \forall i, r, t, \quad (2)$$

where $t_i$ is the set of the fine-grained time slots contained in $l$’s request, $g_{r,k}$ denotes the amount of resource $r$ needed to assemble each type-$k$ VM, and $N^l_{A,i,k}$ is the number of type-$k$ VMs allocated in edge cloud $i$ if the allocation

| Symbol | Explanation |
|--------|-------------|
| $i$    | Edge cloud index |
| $k$    | VM type index |
| $K$    | The number of VM types |
| $l$    | Request index |
| $\mathcal{O}_{l,k}$ | The set of data from request $l$ associated with each type-$k$ VM |
| $L^l$ | The length of service required by $l$ |
| $c_{i,r,t}$ | Available amount of resource $r$ in edge cloud $i$ at time slot $t$ |
| $g_{k,r}$ | Amount of resource $r$ to assemble one type-$k$ VM |
| $A^l$ | Set of feasible resource allocation configuration for request $l$ |
| $N^l_{A,i,k}$ | The number of type-$k$ VMs in edge cloud $i$ with configuration $A$ |
| $p_k$ | Revenue rate for hosting one VM of type $k$ |
| $R(T)$ | The revenue obtained from coarse-grained time slot $T$ |
| $C(T)$ | The data-transportation cost incurred in coarse-grained time slot $T$ |
| $C$ | The predetermined upper bound of the time-average transportation cost |
| $w_{i,j}$ | The latency between edge cloud $i$ and $j$ |
| $Q(T)$ | The virtual queue backlog of coarse-grained time slot $T$ |
| $x^l_{A}$ | 0 – 1 decision variable which is set to 1 if configuration $A$ is adopted for request $l$ |
| $d_{i,o}(T)$ | The aggregated demand from $i$ for data $o$ during coarse-grained time slot $T$ |
| $S_i$ | The placed public data set in edge cloud $i$ |
| $S_t$ | The cache size in edge cloud $i$ |
| $\alpha_i$ | The Lagrangian multiplier corresponding to constraint (1) |
| $\beta_{i,r,t}$ | The Lagrangian multiplier corresponding to constraint (2) |

### TABLE 1: Symbols used in this article
strategy $A \in A^l$ is adopted. This method helps approximate the theoretically competitive solutions. In this work, the requests arrive at arbitrary times, and our method responds immediately rather than buffer the requests until the end of a time slot.

2.3 Revenue

In this work, instead of the revenue during a specific period, we aim to maximize long-term ECP revenue. We present the long-term revenue with a time-average formulation. Specifically, denoting the revenue obtained from a coarse-grained time slot $T$ as $R(T)$, we have

$$R(T) = \sum_{t: T \leq t < T+1} \sum_{A \in A^l} \sum_{k} p_k \sum_{i} N_{A,i,k} x_{A,i,k}^l,$$

where $t^l$ is the arrival time of $l$, and $p_k$ is the price rate for a type-$k$ VM. Given the revenue obtained per time slot, we can define the time-average revenue of the ECP as

$$\overline{R(T)} = \lim_{T \to \infty} \frac{1}{T} \sum_{T=0}^{T-1} R(T).$$

2.4 Data-transportation costs

Inter-cloud data transportation is inevitable, decreasing the quality of service (QoS) due to latency and, even worse, making it expensive for the ECP to use the limited inter-cloud links to transport bulk data. In this article, inter-cloud data-transportation costs occur due to two reasons: 1) when the ECP does not provision the VMs for a mobile user in the edge cloud with which the mobile device is associated, the ECP must transport the user’s uploaded data to the other edge clouds where the VMs are provisioned with the inter-cloud links; 2) if a certain user’s VM cannot find the data to be processed in the local cache, the VM tries to fetch the data from its neighboring edge clouds, which incurs inter-cloud data-transportation costs. In this article, we denote the data-transportation cost for data $o$ with size $s_o$ between the edge clouds $i$ and $j$ as $s_o w_{i,j}$, where $w_{i,j}$ is the latency between $i$ and $j$. We also denote the data cached in edge cloud $i$ as $S_i$ and the dataset required by $l$ associated with each type $k$ VM as $S_{i,k}$. Provided the data requirements and denoting $f_{i,j,o}$ as the amount that $i$ obtains from $j$ for $o$, we can formulate the transportation cost in a coarse-grained time slot $T$ as

$$C(T) = \sum_{t: T \leq t < T+1} \sum_{A \in A^l} \left( \sum_{i} \sum_{o \in S_{i,k} - S_i} w_{i,j} f_{i,j,o} \right) x_{A,i,k}^l.$$

Given that the transportation cost for the unit traffic between locations is fixed, the fractional routing $f_{i,j,o}$ becomes superfluous because the request for object $o$ is always directed to the $j$ with the lowest $w_{i,j}$, that is,

$$j^* = \arg\min \{ w_{i,j} : o \in S_j \}.$$

Thus, (5) becomes:

$$C(T) = \sum_{t: T \leq t < T+1} \sum_{A \in A^l} \left( \sum_{i} \sum_{o \in S_{i,k} - S_i} w_{i,j^*} s_o \right) x_{A,i,k}^l.$$

The transportation cost is inevitable and should be kept under a certain level. However, the transportation-cost constraint is different than resource-capacity constraints in that the resource capacity can never be violated, while for the transportation cost constraint, violation in certain time slots is allowable. In particular, in this article, we consider controlling the time-average transportation cost under a predetermined number $C$. Specifically, we denote the time-average transportation cost as

$$C(T) = \lim_{T \to \infty} \frac{1}{T} \sum_{T=0}^{T-1} C(T),$$

and let

$$C(T) \leq C.$$

With such relaxation, we can better take advantage of the request dynamics.

2.5 Problem formulation

By summarizing the above models, we can define the revenue-maximization problem as follows:

$$\max \overline{R(T)}$$

s.t. (1), (2), and (6).

Notice that the above problem definition has two subproblems to solve: the computing resource-allocation problem and the public data-placement problem. The two problems have different temporal and spatial characteristics. For the temporal aspect, because the request could arrive at an arbitrary time, the ECP must make computing-resource-allocation decisions immediately; however, because the popularity distribution of the data changes relatively slowly, data placement should be determined based on observing a relatively long period, and frequent re-allocation does not help enhance the hit ratio. For the spatial aspect, VM migration between edge clouds is complicated, so if a computing resource-allocation decision is made, it does not change in the future, whereas for data placement, data should be re-allocated to improve the hit rate and thus reduce the data-transportation cost. In the next section, we describe our approach to solve the computing resource-allocation and public data-placement-optimization problems jointly.

3 The online approach to solve the joint optimization problem

In this section, we present our approach to solve the computing resource-allocation and data-placement problems jointly. Our approach works with hybrid timescales: a fine-grained time slot for computing resource allocation and a coarse-grained time slot for public data placement. Fig. 2 shows the framework. For the coarse-grained timescale, we notice that constraint (6) has a time-average formulation, which allows us to trade off the objective function and the constraint violation. For this purpose, we introduce a virtual queue whose length represents the “budget” for transportation costs that can be “consumed” in the next coarse-grained time slot, and then we employ the drift-plus-penalty method to optimize the long-term revenue and stabilize the virtual
queue. Specifically, at the beginning of each coarse-grained time slot, we accumulate the transportation cost from the previous time slot, update the backlog of the virtual queue, and then make computing resource-allocation decisions to optimize the sum of the weighted revenue and the “drift” of the virtual queue.

This drift-plus-penalty-based method has attracted much interest in recent years because it can provide near-optimal solutions without assumptions about future knowledge. However, it usually works in a buffer-and-decision manner, that is, buffering the requests for a certain amount of time and then making a decision on how to deal with the buffered requests. For this reason, the vanilla drift-plus-penalty method works well for delay-tolerant tasks but is incapable of addressing real-time problems. In this work, we overcome the shortcomings of the drift-plus-penalty method by introducing a primal-dual online algorithm to address the real-time requirement of computing resource allocation. Because the vanilla form of the primal-dual algorithm cannot be applied to the scenario of tasks finishing in a finite amount of time, we further discretize the coarse-grained time slot into fine-grained time slots to approximate the results obtained from the vanilla primal-dual algorithm.

When a request arrives, it is responded to in real time. At the end of each coarse-grained time slot, we re-allocate the data among different edge clouds to reflect the data-popularity change in the past time slot. With the proposed method, the immediate effect of computing resource-allocation decisions can be reflected in the long term, and long-term data-allocation decisions can guide computing resource allocation for a relatively long period.

3.1 The online joint optimization framework

We introduce a virtual queue to represent the difference between the actual transportation cost and the predetermined upper bound \( C \). By introducing the virtual queue, the constraint \( \Box \) can be satisfied with queue-stabilization technology. Specifically, we denote the queue backlog of time slot \( T \) as \( Q(T) \), and the dynamics are defined as

\[
Q(T + 1) = \max\{Q(T) + C(T) - C, 0\}
\]

where \( C \) is the predetermined upper bound of the time-average transportation cost. The fact that the virtual queue is stable implies that the constraint \( \Box \) is satisfied. This can be seen from the following lemma:

**Lemma 1.** \( \lim_{T \to \infty} Q(T)/T = 0 \) implies \( \overline{C(T)} \leq C \).

**Proof.** From \( \Box \), we have \( Q(T + 1) = \max\{Q(T) + C(T) - C, 0\} \geq Q(T) + C(T) - C \). Thus, \( Q(T + 1) - Q(T) \geq C(T) - C \) for all \( T \). Use telescoping sums over \( T \in \{0, ..., T - 1\} \), we have \( Q(T) - Q(0) \geq \sum_{T=0}^{T-1} (C(T) - C) \). Dividing by \( T \) and taking the limit as \( T \to \infty \), we have

\[
\frac{C(T) - C}{T} \leq \lim_{T \to \infty} \frac{Q(T)}{T}.
\]

From \( \Box \), we know that the virtual queue rate is stable only if

\[
\lim_{T \to \infty} \frac{Q(T)}{T} \leq 0.
\]

Thus, the lemma is proved.

To stabilize the virtual queue, we define the Lyapunov function as

\[
L(Q(T)) = \frac{1}{2} Q^2(T),
\]

and the 1-slot drift as

\[
\Delta_1(Q(T)) = L(Q(T + 1)) - L(Q(T)).
\]

Following the drift-plus-penalty method \( \Box \), we can make the time-average revenue \( \overline{R(T)} \) within a constant optimality gap and stabilize the virtual queue simultaneously by maximizing a lower bound of the following term in each coarse-grained time slot:

\[
VR(T) - \Delta_1(Q(T)),
\]

where \( V \) is a predetermined non-negative parameter to control the trade-off between revenue and the data-transportation cost violation. We assume that \( C^2(T) \) is deterministically upper-bounded by \( C_{\max}^2 \), a lower bound of the above term can then be obtained from the dynamics of the virtual queue. In particular,

\[
L(Q(T + 1)) - L(Q(T)) = \frac{1}{2} (Q(T + 1) - Q^2(T)) = \frac{1}{2} (\max\{Q(T) + C(T) - C, 0\}^2 - Q^2(T)) \leq Q(T)(C(T) - C) + B,
\]

where \( B = \max\{C_{\max}^2, C^2\} \). Multiplying both sides of the above inequation by \( (-1) \) and adding \( VR(T) \) to each side yields a lower bound of \( \Box \):\n
\[
VR(T) - \Delta_1(Q(T)) \geq VR(T) - Q(T)(C(T) - C) - B.
\]

In each coarse-grained time slot, we greedily maximize the right-hand side of \( \Box \) with constraints \( \Box \) and \( \Box \), and in the long run, we approximate the global optimality and maintain the virtual queue stable. Alg. \( \Box \) summarizes the entire process:

![Fig. 2: Framework of the online mechanism with hybrid timescales](image-url)
Algorithm 1 The online joint optimization framework

Input:
The backlog of the virtual queue: $Q(T)$

Output:
The computing resource allocation decision $\{x_A^t\}$
The public data-placement decision $\{S_i\}$

1: Update the virtual queue in accordance with (10)
2: for each request arriving between $T$ and $T + 1$ do
3: Execute Alg. 2 to decide computing resource allocation
4: end for
5: Compute the aggregated data demand $\{d_{i,o}(T)\}$ at the end of the current time slot
6: Execute Alg. 3 to update the data placement $\{S_i\}$

In the following sections, we present the two sub-routines of this algorithm in detail.

3.2 Online computing-resource allocation

The requests would arrive at arbitrary times, and the original drift-plus-penalty method only works in a buffering-and-deciding manner. In this work, we make a fundamental extension to the drift-plus-penalty algorithm with a primal-dual approach. To explain in detail, we reformulate the optimization problem of a coarse-grained time slot $T$ as follows:

$$
\max_x \sum_{i,T \leq t < T + 1} L_i^t \sum_{A \in A^t} \tilde{R}_A^t x_A^t
$$

s.t.  \[2], \[1]

where

$$
\tilde{R}_A^t = V \sum_k p_k \sum_i N_{A,i,k}^t \sum_{i,k} N_{A,i,k}^t \left( \sum_{o \in D_{i,k} \cup o \in S_i} w_{i,j,s_o / L_i^t} \right)
= \sum_{i,k} N_{A,i,k}^t \left( Vp_k - \sum_{o \in D_{i,k} \cup o \in S_i} w_{i,j,s_o / L_i^t} \right).
$$

For convenience, we denote the “component” of edge cloud $i$ of $\tilde{R}_A^t$ as

$$
\tilde{R}_{A,i}^t = \sum_k N_{A,i,k}^t \left( Vp_k - \sum_{o \in D_{i,k} \cup o \in S_i} w_{i,j,s_o / L_i^t} \right),
$$

and it is clear that $\tilde{R}_A^t = \sum_i \tilde{R}_{A,i}^t$. We introduce the Lagrangian multiplier $\alpha^t$ to constrain [1] and $\beta_{i,r,t}$ to constrain [2], respectively, so we have the dual problem as follows:

$$
\min_{\alpha, \beta} \sum_i \alpha^t + \sum_i \sum_r \sum_t c_{i,r,t} \beta_{i,r,t}
$$

s.t.  \[19\]

$$
\alpha^t \geq L_i^t \tilde{R}_A^t - \sum_{i,r,t,k} N_{A,i,k}^t g_r \beta_{i,r,t} \quad \forall A \in A^t, \forall l
$$

$$
\alpha^t \geq 0, \beta_{i,r,t} \geq 0.
$$

With the Lagrangian multipliers, we propose the following online algorithm to respond to users’ requests:

Algorithm 2 Online computing-resource allocation algorithm

Input:
The backlog $Q(T)$ of the virtual queue
The public data-placement profile $\{S_i\}$
The request sequence (indexed by $l$)

Output: Computing resource-allocation decisions

1: Initialize $\beta_{i,r,t} \leftarrow 0 \forall i, r, t$
2: For each request $l$
3: Compute the $A^* \in A^t$ such that
$$
A^* = \arg \max_{A \in A^t} \left( L_i^t \tilde{R}_A^t - \sum_{i,r,t,k} N_{A,i,k}^t g_r \beta_{i,r,t} \right)
$$
4: if $L_i^t \tilde{R}_A^t - \sum_{i,r,t,k} N_{A,i,k}^t g_r \beta_{i,r,t} < 0$ or $\beta_{i,r,t} > 1$
5: else
6: Allocate resources to $l$ as $A^*$
7: Update $\beta_{i,r,t}$ as
$$
\beta_{i,r,t} \leftarrow \beta_{i,r,t} \left( 1 + \sum_k N_{A,i,k}^t g_r \right) + \frac{1}{e - 1} K_{C,i,r,t}
$$
8: if $L_i^t \tilde{R}_A^t - \sum_{i,r,t,k} N_{A,i,k}^t g_r \beta_{i,r,t}$
9: end if

3.3 Public data placement

Notice that the public data-popularity distribution reflects the relatively long-term aggregating requirements; we do not update the data placement at the arrival of individual requests but at the end of each coarse-grained time slot.

We denote the size of content $o$ as $s_o$ and the size of the storage in edge cloud $i$ as $S_i$. Given the aggregating data demand in time slot $T$, the data-placement problem can be modeled as a transportation cost minimization problem subject to the storage-size constraints. Remember that $d_{i,o}(T)$ is the aggregating data demand in edge cloud $i$ for $o$ during time slot $T$. Clearly,

$$
d_{i,o}(T) = \sum_{T \leq t < T + 1 \in A^t} \sum_k N_{A,i,k}^t x_{A,i}^t \sum_{o \in D_{i,k} \cup o \in S_i} s_o.
$$

For edge cloud $i$, we denote the placed data set as $S_i$. We call a data placement in $i$ feasible if the total size of the placed data does not exceed the storage size, that is,

$$
\sum_{o \in S_i} s_o \leq S_i.
$$

We define the set of all the feasible data-placement profiles of $i$ as $F_i$ and introduce an indicator vector $\{f_1, f_2, \ldots, f_{|F_i|}\}$ to denote whether a feasible data-placement profile is adopted, where $|F_i|$ is the number of all feasible data-placement profiles of $i$. For example, suppose the universe of data as $\{o_1, o_2, o_3\}$ with $s_{o_1} + s_{o_2} \leq S_i$, $s_{o_1} + s_{o_3} \leq S_i$, and $s_{o_1} + s_{o_2} + s_{o_3} > S_i$, so $F_i = \{\{o_1\}, \{o_2\}, \{o_3\}, \{o_1, o_2\}, \{o_1, o_3\}\}$. Only one feasible caching profile in $F_i$ can be selected for a specific time slot. Consider that in a certain coarse-grained time slot, $\{o_1, o_2\}$
Lemma 2. The public data-placement problem defined with (22) and (23) is NP-hard.

The above data-placement problem can be reduced to a general assignment problem (GAP), so it is NP-hard. Before presenting our approximation algorithm, we introduce some important concepts needed for further instruction [5].

Definition 1 (Submodular function). Let $X$ be a finite set and the function $f : 2^X \to M$. $f$ is submodular if for all $A, B \subseteq X$, $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$.

An equivalent definition of “submodular” is based on marginal value. Denoting the marginal value of $i$ with respect to $A$ as $f_A(i) = f(A + i) - f(A)$, if $\forall A \subseteq B \subseteq X$ and $\forall i \in A - B$, $f_A(i) \leq f_B(i)$, then $f$ is supermodular.

Definition 2 (Simple Partition Matroid). $X$ is a ground set partitioned into $l$ disjoint sets $X_1 \cup X_2 \cup \ldots \cup X_l$ with associated integers $k_1, k_2, \ldots, k_l$, and $I = A \subseteq X : |A \cap X_i| \leq k_i, i = 1, \ldots, l$. Then $M = (X, I)$ is a partition matroid. Specifically, if $k_i = 1 \forall i$, the matroid is called a “simple partition matroid.”

We define the ground set $X$ as $\{(i, F)|F \in \mathcal{F}_i\}$. Letting $I = \{(i, F_i) \in X : F_i \in \mathcal{F}_i\}$, we can reformulate the data placement-optimization problem as follows:

$$
\min_{(i, F_i)} f((i, F_i)) = \sum_{t:T \leq t < T+1} \sum_{i} d_{i,o}(T) w_{i,j,*}, \quad (24)
$$

$$
s.t. \{(i, F_i)\} \in I. \quad (25)
$$

Lemma 3. The function $f((i, F_i))$ is a monotone supermodular function.

Lemma 4. The constraint (25) corresponds to a simple partition matroid $M = (X, I)$.

The proofs of Lemma 3 and 4 are similar to the proofs of Lemma 3 and 2 of [6]. Thus, we establish that the data-placement problem is a minimization of a monotone supermodular function subject to simple partition matroid constraints. Therefore, a greedy algorithm, like the one shown in Alg. 3, obtains a $1/2$-competitive solution.

Algorithm 3 Data-placement algorithm

**Input:**
- The aggregating data requirement $\{d_{i,o}(T)\}$
- The size of storage $\{S_i\}$
- The size of each data $\{s_o\}$

**Output:**
- The data-placement decision $S$
- Initialize $S \leftarrow \emptyset$; $\mathcal{I} \leftarrow \{i\}$

1. **while** $\mathcal{I} \neq \emptyset$ **do**
2. **for** $i \in \mathcal{I}$ **do**
3. $F_i^* \leftarrow \arg\max_{F_i} f_{\mathcal{I}}(F_i)$
4. **end for**
5. $F^* \leftarrow \max_{i \in \mathcal{I}} \{F_i^*\}$
6. $S \leftarrow S \cup F^*$
7. $\mathcal{I} \leftarrow \mathcal{I} \setminus \{i : F_i^* = F^*\}$
8. **end while**

Notice that we use $f_{\mathcal{I}}(F_i)$ to denote the marginal benefit of adding $F_i$ to the already-chosen set $S$.

4 Performance analysis

In this section, we present a performance analysis of the proposed method. We first analyze the performance of the computing resource-allocation algorithm and the data-placement algorithm within a coarse-grained time slot and then present the long-term performance guarantee of the ECP’s revenue.

4.1 The optimality within a coarse-grained time slot

We begin analyzing online computing-resource allocation for a certain coarse-grained time slot $T$ with the following claim.

**Proposition 1.** Alg. 2 returns an integral solution and has the competitive ratio $1 - 1/e$.

We prove this claim by showing the feasibility of the dual and primal problems with Lemma 5 and 6 and the primal-to-dual ratio with Lemma 7.

**Lemma 5.** Alg. 2 produces a feasible dual solution.

**Proof.** Consider a dual constraint corresponding to request $l$.

$$
L^l \tilde{R}^*_A - \sum_{t,r,t,k} N^l_{\ast,i,t,k} g_{r,t,k} \beta_{i,r,t} < 0,
$$

the dual constraint cannot be satisfied. Otherwise, the algorithm allocates the resources to $l$ as $A^*$. Setting

$$
\alpha_l = L^l \tilde{R}^*_A - \sum_{t,r,t,k} N^l_{\ast,i,t,k} g_{r,t,k} \beta_{i,r,t}
$$

guarantees that the constraint is satisfied for all $l$.

**Lemma 6.** Denoting $\delta^p$ and $\delta^d$ the changes in the primal and dual costs for iterating Algorithm 2 in each iteration, $\delta^p = (1 - 1/e)\delta^d$.

**Proof.** Whenever the algorithm updates the primal solutions, the change in the objective function is

$$
\delta^p = L^l \tilde{R}^*_A.
$$
Remind that \( A^* \) denotes the allocation for request \( l \). Then the change in the dual objective function has two parts: the change corresponding to \( \alpha^l \) and the change corresponding to \( \beta_{i,r,t} \). Denoting the former as \( \delta^D_\alpha \) and the latter as \( \delta^D_\beta \), we have

\[
\delta^D_\alpha = L^i \tilde{R}_{A^*} - \sum_{i,r,t,k} N^l_{A^*,i,k} g_{r,k} \beta_{i,r,t},
\]

and

\[
\delta^D_\beta = \sum_{i,r,t} c_{i,r,t} \left( \sum_k N^l_{A^*,i,k} g_{r,k} \beta_{i,r,t,k} + \frac{1}{e - 1} \tilde{R}_{A^*,i} \right).
\]

Then we have

\[
\delta^D = \delta^D_\alpha + \delta^D_\beta
\]

\[
= L^i \tilde{R}_{A^*} + \sum_{i,r,t,k} \frac{1}{e - 1} \tilde{R}_{A^*,i}
\]

\[
= L^i \tilde{R}_{A^*} + \frac{1}{e - 1} L^i \tilde{R}_{A^*}
\]

\[
= \left(1 + \frac{1}{e - 1}\right) \delta^P.
\]

Thus, the lemma is proved.

**Lemma 7.** Algorithm 2 produces an almost-feasible primal solution in which the resource constraint \( c_{i,r,t} \) is violated by at most \( \max_{i,r,t} \sum_k N^l_{A^*,i,k} g_{r,k} \).

**Proof.** Consider a primal constraint \( \sum_{k \in K} \sum_{A \in A^l} N^l_{A,i,k} x_A^l \leq c_{i,r,t} \). Whenever we increase some \( x_A^l \) by 1, we increase \( \beta_{i,r,t} \) by some factor. The value of \( \beta_{i,r,t} \) behaves like a geometric sequence. Formally, we claim that

\[
\beta_{i,r,t} \geq \frac{e \sum_k N^l_{A^*,i,k} g_{r,k} - 1}{e - 1}.
\]

This can be proved by induction. Note that \( \beta_{i,r,t} = 0 \) initially, so the statement is trivially true. We assume that the inequality is satisfied up to request \( l - 1 \), that is,

\[
\beta_{i,r,t} < \frac{e \sum_{i,k} N^l_{A^*,i,k} g_{r,k} - 1}{e - 1}.
\]

After the arrival of \( l \),

\[
\beta_{i,r,t}^{< \text{bef}>} = \beta_{i,r,t}^{< \text{bef}>} + \frac{1 + \sum_k N^l_{A^*,i,k} g_{r,k} - 1}{e - 1} \tilde{R}_{A^*,i} + \frac{1}{e - 1} K c_{i,r,t}
\]

\[
\geq \left(1 + \frac{1}{e - 1} \tilde{R}_{A^*,i} \right) \left(1 + \sum_k N^l_{A^*,i,k} g_{r,k} \right)
\]

\[
= \left(1 + \frac{1}{e - 1} \tilde{R}_{A^*,i} \right) \left(1 + \sum_k N^l_{A^*,i,k} g_{r,k} \right) + \frac{1}{e - 1} \tilde{R}_{A^*,i} + \frac{1}{e - 1} K c_{i,r,t}
\]

\[
\geq \frac{e \sum_{i,k} N^l_{A^*,i,k} g_{r,k} - 1}{e - 1}.
\]

The first inequality follows the updating rules of \( \beta_{i,r,t} \) of Alg. 2. The second inequality follows the fact that

\[
1 + \sum_k N^l_{A^*,i,k} g_{r,k} \leq c_{i,r,t}.
\]

given that \( \sum_k N^l_{A^*,i,k} g_{r,k} / c_{i,r,t} \) approaches 0. The third inequality is achieved by scaling \( p_k \) so that \( \tilde{R}_{A^*,i} \geq K \sum_k N^l_{A^*,i,k} g_{r,k} \).

Observe that \( \sum_{k \in K} \sum_k N^l_{A^*,i,k} g_{r,k} \geq c_{i,r,t} \) implies that \( \beta_{i,r,t} \geq 1 \), in which case we reject the request, so the resource constraint \( c_{i,r,t} \) is violated by at most \( \max_{i,r,t} \sum_k N^l_{A^*,i,k} g_{r,k} \) which is far less than \( c_{i,r,t} \).

**4.2 Optimality of public data placement**

**Proposition 2.** The proposed public data-placement algorithm (i.e., Alg. 3) achieves a competitive ratio of 1/2.

This emerges from the proposed algorithm greedily optimizing a submodular monotone set function over constraints associated with a matroid.

**Optimality against the N-slot Look-ahead Mechanism**

Because the request arrival rates and the data popularities are arbitrary, finding the global optimal solutions is difficult. Instead of comparing with the optimal result directly, we introduce an \( N \)-slot look-ahead mechanism as an approximation of the global optimal revenue. The \( N \)-slot look-ahead mechanism has the same objective function as (9) but assumes that the request sequence and an approximation of the global optimal revenue. The

\[
\sum_{s=1}^{l} \sum_k N^s_{A^*,i,k} g_{r,k} \geq c_{i,r,t}
\]

implies that \( \beta_{i,r,t} \geq 1 \), in which case we reject the request. In particular, in the \( N \)-slot look-ahead mechanism, time is divided into frames, each frame consisting of \( N \) coarse-grained time slots. Suppose the \( z \)th time frame consists of the following coarse-grained time
slots: \( \{zN, zN + 2, \ldots, zN + N - 1\} \). In each time frame, the following problem must be solved:

\[
\max \frac{1}{N} \sum_{T=zN}^{zN+N-1} R(T) \quad \text{s.t.} \quad (1), \ (2), \ \text{and} \ \frac{1}{N} \sum_{T=zN}^{zN+N-1} C(T) \leq C,
\]

where \( z = 0, 1, \ldots \). The \( N \)-slot look-ahead mechanism assumes that all the request arrivals in the coming \( N \) coarse-grained time slots are known in advance and that the data popularity estimation is perfect, thus approximating the global optimal solution. We show that the time-average revenue \( \bar{R}(T) \) yielded by the proposed method has a constant gap from the result of the \( N \)-slot look-ahead mechanism.

**Theorem 1.** Let \( \bar{R}_N(z) \) denote the optimal objective function value with the \( N \)-slot look-ahead problem in the \( z \)-th time frame. Suppose a period of \( 2N \) coarse-grained time slots, where \( Z \) is a constant. We have

\[
\frac{1}{Z} \sum_{T=0}^{2N-1} R(T) \geq \left(1 - \frac{1}{e}\right) \left(\frac{1}{Z} \sum_{z=0}^{N-1} \bar{R}_N(z) - B N V\right).
\]

**Proof.** Remember that for the 1-slot drift, we have

\[
\Delta_1(T) = L(Q(T + 1)) - L(Q(T)) \leq Q(T)(C(T) - (C)) + B,
\]

where \( B = \max(|C(\max)|^2, c^2) \). We multiply both sides by \( -1 \) and add \( V R(T) \) to each side:

\[
VR(T) - \Delta_1(T) \geq VR(T) - Q(T)(C(T) - (C)) - B \geq \left(1 - \frac{1}{e}\right) \left(V R^*(T) - Q(T)(C^*(T) - (C)) - B\right),
\]

where \( R^*(T) \) is the revenue and \( C^*(T) \) is the data-transportation cost from coarse-grained time slot \( T \) from any alternative resource-allocation decision. The last inequality emerges from the proposed method trying to maximize \( VR(T) - Q(T)(C(T) - (C)) - B \) for every \( T \) with the primal-dual online algorithm, yielding a \((1 - \frac{1}{e})\)-competitive solution (Proposition 4).

Now we consider a process starting from coarse-grained time slot \( zN + n \), where \( n = 0, \ldots, N - 1 \). We have

\[
VR(zN + n) - \Delta_1(zN + n) \geq \left(1 - \frac{1}{e}\right) \left(V R^*(zN + n) - Q(zN + n)(C^*(zN + n) - (C)) - B\right).
\]

Considering that

\[
|Q(zN + n) - Q(zN)| \leq n \max\{|C(\max)|^2, c^2\} = 2n B,
\]

we can substitute it into (30), and it follows that

\[
VR(zN + n) - \Delta_1(zN + n) \geq \left(1 - \frac{1}{e}\right) \left(V R^*(zN + n) - Q(zN)(C^*(zN + n) - (C)) - B - 2n B\right).
\]

We sum up the above inequality with \( n = 0, \ldots, N - 1 \) and denote the \( N \)-slot drift as

\[
\Delta_N(zN) = L(Q(zN + N)) - L(Q(zN)),
\]

We have

\[
V \sum_{T=zN}^{zN+N-1} R(T) - \Delta_N(zN) \geq \left(1 - \frac{1}{e}\right) \left(V \sum_{T=zN}^{zN+N-1} R^*(T) - Q(zN) \sum_{T=zN}^{zN+N-1} (C^*(T) - (C)) - \sum_{T=zN}^{zN+N-1} (B + 2n B)\right) = (1 - \frac{1}{e}) \left(V \sum_{T=zN}^{zN+N-1} R^*(T) - Q(zN) \sum_{T=zN}^{zN+N-1} (C^*(T) - (C) - B N^2)\right).
\]

The last equation emerges from

\[
\sum_{z=0}^{N-1} n = \frac{N(N + 1)}{2}.
\]

We can replace \( \sum_{T=zN}^{zN+N-1} R^*(T) \) with \( \bar{R}_N(z) \) in the above equations:

\[
V \sum_{T=zN}^{zN+N-1} R(T) - \Delta_N(zN) \geq \left(1 - \frac{1}{e}\right) \left(V N \sum_{z=0}^{N-1} \bar{R}_N(z) - B N^2\right)\).
\]

Notice that in the above inequality, we eliminate the term \( Q(zN) \sum_{T=zN}^{zN+N-1} (C^*(T) - (C)) \) because, with the \( N \)-slot look-ahead mechanism, \( Q(zN) \sum_{T=zN}^{zN+N-1} (C^*(T) - (C)) \leq 0 \) deterministically. We sum up the above equation over \( z \in {0, \ldots, Z - 1} \) as follows:

\[
V \sum_{T=0}^{Z-N-1} R(T) - (L(Q(N)) - L(Q(0))) \geq \left(1 - \frac{1}{e}\right) \left(V N \sum_{z=0}^{Z-1} \bar{R}_N(z) - B Z N^2\right).
\]

Dividing both sides of the above inequality by \( V N Z \), we prove the theorem. \( \square \)

## 5 Performance Evaluation

**Basic settings.** To evaluate the proposed method, we developed a discrete-event simulator with MATLAB. In this section, we present the evaluation results with simulations. We consider an ECP of moderate scale that operates 5 edge clouds. Each cloud has three kinds of resources (e.g., CPU, memory, storage, etc.) for mobile-task computing. To keep the research general, we do not specify the resources. We assume the resources are divisible and the initial amount of each resource is 5,000 in each location. The resources can be used to resemble two types of VMs. Type-1 VMs need 10,
20, and 30 of each resource, while type-2 VMs need 30, 20, and 10. The price of a type-1 VM is 10 per unit time, and the price of a type-2 VM is 20. Besides the resources needed to assemble VMs, we also assume that there are caching devices in each edge cloud that can transparently cache popular data. The size of each caching device is the same in each edge datacenter. Without specific illustration, the total cache size is 40% the size of the universal content. In this research, we consider not only the data that can be obtained from the Internet (public data) but also the data uploaded by mobile users (private data). Without specific illustration, we set the total amount of private data as twice the amount of public data. The edge clouds are interconnected with high-speed networks. The end-to-end latency of fetching data from the local cache, from a neighboring cache, and from the remote cloud is randomly assigned following uniform distributions in the ranges $[5, 10]$,$[20, 50]$,$[100, 200]$, respectively [7]. We discretize time with coarse-grained time slots and fine-grained time slots, with each coarse-grained time slot containing 500 fine-grained time slots. The lifetime of each VM is assumed to be uniformly distributed in the range $[1, 5]$ of fine-grained time slots. The data popularity follows a Zipf distribution with an exponent of 0.6 [7]. In the rest of this section, we compare the proposed method with some benchmark methods and discuss the impact of the parameters.

**Evaluation with dynamics of VM request.** In this simulation, we let requests follow Poisson arrivals; moreover, the expected arrival speed $\lambda$ varies randomly from 0 to 50 every 25 fine-grained time slots. Type-1 and type-2 VMs are requested with equal probability. Fig. 3 shows the dynamics of the requests. During the simulation, we kept the data popularity fixed. We implemented two methods as benchmarks: myopic resource allocation with cooperative caching (MyopicCoop) and myopic resource allocation with non-cooperative caching (MyopicNoCoop). Myopic resource allocation refers to the method that, when an ECP receives a request, it assembles the required VMs with the minimum transportation cost. With cooperative caching, the ECP makes caching decisions with Alg. 3 while with non-cooperative caching, each caching device caches the data with the highest popularities independently. We set the time-average transportation cost constraint $L = 35,000$ and the drift-plus-penalty related parameter $V = 100,000$. Fig. 4 compares the time-average revenues of the proposed method and the benchmark methods. Fig. 5 shows the transportation costs of the three methods. We can see that the proposed method performs much better than the myopic methods while satisfying the long-term transportation cost constraint. Fig. 5 partly illustrates the advantage of the proposed method: taking better advantage of the time-average expression of the transportation cost constraint. In some coarse-grained time slots, even if the transportation cost greatly exceeds $L$, the requests can still be accepted. By doing so, in the successive time slot, the relative weight of the transportation cost increases relative to revenue. Intuitively, the proposed method has a larger feasible region given the time-average expression of the transportation cost constraint than the sample myopic methods. To study the stability of the system, we also checked the length of the virtual queue during the simulation period, as shown in Fig. 6, which shows, which approaches 0 over time.

**Evaluation with dynamics of data popularity.** In this research, the dynamics depend not only on VM requests but also on data popularity. The data popularity changes with time, and as far as we know, no effective method exists to formally define and measure the popularity change. In this experiment, we propose a method that avoids directly measuring the popularity change of each data; instead, we introduce the concept “popularity estimation error rate.”
The estimation error rate is a Poisson random number with an average value of 0.3, which means that the probability of the data contributing to the top 50% of the traffic is not cached. Fig. 7 shows the dynamics of the estimation error. In this experiment, we fix the request-arrival rate at 25 per fine-grained time slot to focus on the impact of data-popularity dynamics. We set the transportation cost $L = 35,000$, and the drift-plus-penalty related parameter $V = 100,000$. Like the earlier experiment, we compare the time-average revenues in Fig. 8, the transportation costs in Fig. 9, and the virtual queue lengths in Fig. 10. Fig. 9 shows larger feasible regions for the optimization problem. In this experiment, we set the ratio of the total cache size to the universal data size to be 0.1, 0.5, and 0.9, the time-average transportation constraint $L = 35000$, and the drift-plus-penalty related parameter $V = 100,000$, and we carry out the simulation for 150 coarse-grained time slots. Fig. 11, 12, and 13 show the time-average revenues, the transportation costs, and the time-average virtual queue lengths, respectively. Fig. 11 intuitively shows the impact of cache size on the transportation costs of the three methods. In Fig. 8, we can see that the proposed method has the best performance. Fig. 9 shows the reason: The proposed method takes better advantage of the time-average constraint expression, while the myopic methods cannot. Fig. 10 shows that the length of the virtual queue approaches 0 over time, implying the stability of the system.

**Evaluation of the impact of total cache size.** The cache size deeply influences the system. Larger cache sizes allow more data to be cached, therefore incurring smaller transportation costs. Mathematically, larger cache sizes produce larger feasible regions for the optimization problem. In this experiment, we set the ratio of the total cache size to the universal data size to be 0.1, 0.5, and 0.9, the time-average transportation constraint $L = 35000$, and the drift-plus-penalty related parameter $V = 100,000$, and we carry out the simulation for 150 coarse-grained time slots. Fig. 11, 12, and 13 show the time-average revenues, the transportation costs, and the time-average virtual queue lengths, respectively. Fig. 11 intuitively shows the impact of cache size on the transportation costs of the three methods. In Fig. 8, we can see that the proposed method has the best performance. Fig. 9 shows the reason: The proposed method takes better advantage of the time-average constraint expression, while the myopic methods cannot. Fig. 10 shows that the length of the virtual queue approaches 0 over time, implying the stability of the system.

**Evaluation of the impact of total cache size.** The cache size deeply influences the system. Larger cache sizes allow more data to be cached, therefore incurring smaller transportation costs. Mathematically, larger cache sizes produce larger feasible regions for the optimization problem. In this experiment, we set the ratio of the total cache size to the universal data size to be 0.1, 0.5, and 0.9, the time-average transportation constraint $L = 35000$, and the drift-plus-penalty related parameter $V = 100,000$, and we carry out the simulation for 150 coarse-grained time slots. Fig. 11, 12, and 13 show the time-average revenues, the transportation costs, and the time-average virtual queue lengths, respectively. Fig. 11 intuitively shows the impact of cache size on the transportation costs of the three methods. In Fig. 8, we can see that the proposed method has the best performance. Fig. 9 shows the reason: The proposed method takes better advantage of the time-average constraint expression, while the myopic methods cannot. Fig. 10 shows that the length of the virtual queue approaches 0 over time, implying the stability of the system.

**Evaluation of the impact of total cache size.** The cache size deeply influences the system. Larger cache sizes allow more data to be cached, therefore incurring smaller transportation costs. Mathematically, larger cache sizes produce larger feasible regions for the optimization problem. In this experiment, we set the ratio of the total cache size to the universal data size to be 0.1, 0.5, and 0.9, the time-average transportation constraint $L = 35000$, and the drift-plus-penalty related parameter $V = 100,000$, and we carry out the simulation for 150 coarse-grained time slots. Fig. 11, 12, and 13 show the time-average revenues, the transportation costs, and the time-average virtual queue lengths, respectively. Fig. 11 intuitively shows the impact of cache size on the transportation costs of the three methods. In Fig. 8, we can see that the proposed method has the best performance. Fig. 9 shows the reason: The proposed method takes better advantage of the time-average constraint expression, while the myopic methods cannot. Fig. 10 shows that the length of the virtual queue approaches 0 over time, implying the stability of the system.
transportation cost is satisfied for all three cache sizes, and Fig. 13 implies the stability of the system.

**Evaluation of the impact of data sources.** Besides the cache size, the ratio of the private-data volume to the public-data volume is also a key factor in the performance of the system. This ratio reflects the composition of the edge applications. An example of private data is a video uploaded from a mobile device for further processing; such private data cannot be shared with others, so caching them has no value. An example of public data is a short video from over-the-top (OTT) providers. A large portion of public data increases the usage of the caching device. In this simulation, we set the ratios of private to public data volume at 0.5, 2.0, and 3.5, the time-average transportation constraint \( L = 35,000 \), and the drift-plus-penalty related parameter \( V = 100,000 \), and we carry out the simulation for 150 coarse-grained time slots. Fig. 14, 15, and 16 show the time-average revenues, the transportation costs, and the time-average virtual-queue lengths. The results are similar to the results with difference cache sizes because more public data implies that more data can be cached, thereby increasing the feasible region of the optimization problem. Fig. 15 and 16 show that the proposed method is feasible and stable with different data compositions.

**Comparison with the \( N \)-look-ahead algorithm** Remem-

ber that we have proved that our method achieves competitive optimal solutions, which means that the performance gap between the proposed method and the theoretically optimal method is fixed. Because it needs vast computing resources to obtain the theoretically optimal results, in this experiment, we implemented an \( N \)-slot look-ahead algorithm to approximate the optimal solution. The algorithm assumes that the future with \( N \) coarse-grained time slots can be perfectly predicted and then solves the following optimization problem. Clearly, if \( N \) approaches infinity, the \( N \)-look-ahead algorithm becomes the theoretically optimal solution. Fig. 17, 18, and 19 show the comparison results with \( N = 5 \).

**Evaluation of the impact of \( V \).** In our algorithm, there is only one parameter, the drift-plus-penalty parameter \( V \), to trade off between revenue and cost. In this experiment, we vary the time-average cost constraint \( L \) from 30,000

\[ \begin{align*}
 &L = 35,000 \\
 &V = 100,000
\end{align*} \]
to 36,000 and V from 60,000 to 180,000 and carried out the simulation 1,000 times, each simulation running for 150 coarse-grained time slots. We show the impact of V in Fig. 20, 21, and 22. Clearly, V controls the trade-off between average acceptance rate, time-average revenue, and time-average cost.

![Graph of Time-average revenues with different L and V](image)

**Fig. 20:** Time-average revenues with different L and V

![Graph of Time-average transportation costs with different L and V](image)

**Fig. 21:** Time-average transportation costs with different L and V

![Graph of Acceptance rates with different L and V](image)

**Fig. 22:** Acceptance rates with different L and V

revenue and the violation of the data-transportation cost constraint in Fig. 20 and 21. Given a certain expectation of the long-term data-transportation cost constraint C, a smaller V would help maintain the constraint while a larger V would violate it. A larger V, however, usually brings more revenue to the ECP than a smaller one, primarily from the request acceptance rate visible in Fig. 22. A larger V usually leads to a higher acceptance rate than a smaller V given the same C. This fact is important for the ECP to determine both C and V in practice to make a better tradeoff between revenue and data-transportation cost.

### 6 RELATED RESEARCH

In recent years, edge computing has increasingly attracted interest. In [9], the authors investigated the energy saving issues, and presented a cooperative fog computing system to process offloading workload on the entire fog layer nodes. They then defined the problem as a joint optimization problem of QoE and energy, and provided optimal solution with low overhead. In [10], the authors discussed the economic issues in large-scale edge computing systems, and proposed a multiattribute-based double auction mechanism in vehicular fog computing to provide efficient and fair resource allocation to vehicular users. In [11], the authors studied the computation-offloading and data routing problems to minimize the computing and energy consumption. They presented a multi-hop cooperative-messaging mechanism that can a stable performance gain for IoT systems. In addition to computing resource allocation, data caching is also a hot topic in edge computing. In [12], the authors studied energy-efficient proactive in-network caching for edge computing, and presented a hybrid edge caching scheme considering the popularity of cached files. The proposed method was proved to be energy efficient with good QoS.

In [13], the authors studied the unique problem of caching fairness in edge computing systems, and presented an approximation algorithm that can significantly improve the data caching fairness. In [14], the authors investigated the multiple bitrate video caching problem, and proved the NP-hardness of the problem. They then presented a 1/2-competitive algorithm with low complexity. Our work is different from most existing research. This article is the first to introduce the general-purpose edge computing, and considers joint optimization of computing resource allocation and data placement in an online manner with provable performance. The primary idea was presented in our previous work [15], but is extensively extended in this article.

In terms of methodology, a great challenge in the resource allocation of edge computing is the high dynamic. As a robust method with low complexity, Lyapunov optimization-based drift-plus-penalty algorithms have attracted interest. Because this method can obtain near-optimal performance without any knowledge of the future, the vanilla algorithm and its variations are widely used in dynamic systems [16] [17] [18] [19]. In [20] and [21], the authors presented applications with two time scales, widely extending the methods usage. However, most Lyapunov optimization-based methods need buffering the 2017 request and therefore cannot satisfy real-time requirements. In [22], the authors proposed a proximate-optimization algorithm that makes online decisions, and the authors of [23] employed the algorithm for resource allocation in distributed clouds. These algorithms are sensitive to real-time constraint violations and so cannot make trade-offs among successive time slots. In our work, we substantially extend existing online algorithms to enable arbitrary time scales for the joint optimization framework in edge computing.

### 7 CONCLUSIONS

Edge computing is increasingly important in the cloud and mobile computing era. By pushing services and data to the edge, network providers release pressure from their core networks, mobile users benefit from enhanced performance,
and service providers have more opportunities to develop emerging services and applications. However, nowadays, a majority of research is focusing on application-specific edge computing systems, which can not take the advantages of edge computing sufficiently. In this work, we present the vision of general-purpose edge computing, and analyze the applications and the ecosystem in general-purpose edge computing environment. We identify the main challenges to realize efficient general-purpose edge computing: interaction of resource allocation and data placement with dynamic requests in heterogenous environment, and then propose a novel online framework and algorithms that can obtain an approximation to the optimality with constant gap without any assumptions and knowledge of the future system states. Both theoretical analysis and simulation results show that with the proposed method, the edge cloud provider can receive near-optimal benefit without assumptions and system future knowledge.

ACKNOWLEDGMENTS

The research is partly supported by the ROIS National Institute of Informatics Open Collaborative Research 2021-21FA03, the Telecommunications Advancement Foundation, Japan, the Cooperative Research Project Program of the Research Institute of Electrical Communication, Tohoku University, and JSPS KAKENHI with the grant number 20H04174.

REFERENCES

[1] W. Cai, R. Shea, C. Huang, K. Chen, J. Liu, V. C. M. Leung, and C. Hsu, “A survey on cloud gaming: Future of computer games,” IEEE Access, vol. 4, pp. 7605–7620, 2016.

[2] W. Cai, F. Chi, X. Wang, and V. C. M. Leung, “Toward multiplayer cooperative cloud gaming,” IEEE Cloud Computing, vol. 5, no. 5, pp. 70–80, 2018.

[3] W. Cai, Y. Chi, C. Zhou, C. Zhu, and V. C. M. Leung, “Ubegaming: Ubiquitous cloud gaming system,” IEEE Systems Journal, vol. 12, no. 3, pp. 2483–2494, 2018.

[4] M. J. Neely, Stochastic Network Optimization with Applications to Communication and Queueing Systems. Morgan and Claypool Publishers, 2010.

[5] G. Calinescu, C. Chekuri, M. Pal, and J. Vondrak, “Maximizing a Monotone Submodular Function Subject to a Matroid Constraint,” SIAM Journal on Computing, vol. 40, no. 6, pp. 1740–1766, 2011.

[6] X. Shao, Z. Liu, M. Dong, H. Masui, and Y. Ji, “A Competitive Approximation Algorithm for Data Allocation Problem in Heterogeneous Mobile Edge Computing,” in Proc. IEEE VTC 2015 Spring, May 2015, pp. 1–5.

[7] T. X. Tran and D. Pompili, “Adaptive Bitrate Video Caching and Processing in Mobile-Edge Computing Networks,” IEEE Transactions on Mobile Computing, vol. 18, pp. 1965–1978, Sept. 2019.

[8] X. Tao, K. Ota, M. Dong, H. Qi, and K. Li, “Performance guaranteed computation offloading for mobile-edge cloud computing,” IEEE Wireless Communications Letters, vol. 6, no. 6, pp. 774–777, 2017.

[9] Y. Dong, S. Guo, J. Liu, and Y. Yang, “Energy-efficient fair cooperation fog computing in mobile edge networks for smart city,” IEEE Internet of Things Journal, vol. 6, no. 5, pp. 7543–7554, 2019.

[10] X. Peng, K. Ota, and M. Dong, “Multiattribute-based double auction toward resource allocation in vehicular fog computing,” IEEE Internet of Things Journal, vol. 7, no. 4, pp. 3094–3103, 2020.

[11] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, “Multi-hop cooperative computation offloading for industrial iot-edge–cloud computing environments,” IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 12, pp. 2759–2774, 2019.

[12] J. Xu, K. Ota, and M. Dong, “Energy Efficient Hybrid Edge Caching Scheme for Tactile Internet in 5G,” IEEE Transactions on Green Communications and Networking, vol. 3, no. 2, pp. 483–493, Jun. 2019.

[13] Y. Huang, X. Song, F. Ye, Y. Yang, and X. Li, “Fair and efficient caching algorithms and strategies for peer data sharing in pervasive edge computing environments,” IEEE Transactions on Mobile Computing, vol. 19, no. 4, pp. 852–864, 2020.

[14] Z. Qu, B. Ye, B. Tang, S. Guo, S. Lu, and W. Zhuang, “Cooperative caching for multiple bitrate videos in small cell edges,” IEEE Transactions on Mobile Computing, vol. 19, no. 2, pp. 289–299, 2020.

[15] X. Shao, G. Hasegawa, N. Kamiyama, Z. Liu, H. Masui, and Y. Li, “Joint optimization of computing resources and data allocation for mobile edge computing (meC): An online approach,” in 2019 28th International Conference on Computer Communication and Networks (ICCCN), 2019, pp. 1–9.

[16] E. Yeh, T. Ho, Y. Cui, R. Liu, M. Burd, and D. Leong, “VIP: A Framework for Joint Dynamic Forwarding and Caching in Named Data Networks,” in Proc. ACM ICN 2014, Sep. 2014, pp. 117–126.

[17] X. Qiu, H. Li, C. Wu, Z. Li, and F. C. M. Lau, “Cost-Minimizing Dynamic Migration of Content Distribution Services into Hybrid Clouds,” IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 12, pp. 3330–3345, Dec. 2015.

[18] Y. Mao, J. Zhang, and K. B. Letaief, “Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices,” IEEE Journal on Selected Areas in Communications, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[19] W. Li, X. Zhou, K. Li, H. Qi, and D. Guo, “TrafficShaper: Shaping Inter-Datacenter Traffic to Reduce the Transmission Cost,” IEEE Transactions on Networking, vol. 26, no. 3, pp. 1193–1206, Jun. 2018.

[20] Y. Yao, L. Huang, A. B. Sharma, L. Golubchik, and M. J. Neely, “Power Cost Reduction in Distributed Data Centers: A Two-Time-Scale Approach for Delay Tolerant Workloads,” IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 1, pp. 200–211, Jan. 2014.

[21] W. Deng, F. Liu, H. Jin, and C. Wu, “SmartDPSS: Cost-Minimizing Multi-Source Power Supply for Datacenters with Arbitrary Demand,” in Proc. IEEE ICDCS 2013, Jul. 2013, pp. 1–13.

[22] N. Buchbinder and J. S. Naor, “Online Primal-Dual Algorithms for Covering and Packing,” Mathematics of Operations Research, vol. 34, no. 2, Apr. 2009.

[23] F. Hao, M. Kodialam, T. V. Lakshman, and S. Mukherjee, “Online Allocation of Virtual Machines in a Distributed Cloud,” IEEE Transactions on Networking, vol. 25, no. 1, pp. 238–249, Jul. 2016.

Xun Shao received his Ph.D. in information science from the Graduate School of Information Science and Technology, Osaka University, Japan, in 2013. From 2013 to 2017, he was a researcher with the National Institute of Information and Communications Technology (NICT) in Japan. Currently, he is an Assistant Professor at the School of Regional Innovation and Social Design Engineering, Kitami Institute of Technology, Japan. His research interests include distributed systems and networking. He is a member of the IEEE and IEICE.

Go Hasegawa received his M.E. and D.E. in information and computer sciences from Osaka University, Japan, in 1997 and 2000, respectively. From July 1997 to June 2000, he was a Research Assistant at the Graduate School of Economics, Osaka University. From 2000 to 2018, he was an Associate Professor at the Cybermedia Center, Osaka University. He is now a professor at the Research Institute of Electrical Communication, Tohoku University. His research work is in information network architecture. He is a member of the IEEE and IEICE.
Mianxiong Dong received B.S., M.S. and Ph.D. in Computer Science and Engineering from The University of Aizu, Japan. He is Vice President and the youngest ever Professor of Muroran Institute of Technology, Japan. He was a JSPS Research Fellow with School of Computer Science and Engineering, The University of Aizu, Japan and was a visiting scholar with BBCR group at the University of Waterloo, Canada supported by JSPS Excellent Young Researcher Overseas Visit Program from April 2010 to August 2011. Dr. Dong was selected as a Foreigner Research Fellow (a total of 3 recipients all over Japan) by NEC C&C Foundation in 2011. He is the recipient of IEEE TCSC Early Career Award 2016, IEEE SCSTC Outstanding Young Researcher Award 2017, The 12th IEEE ComSoc Asia-Pacific Young Researcher Award 2017, Funai Research Award 2018 and NISTEP Researcher 2018 (one of only 11 people in Japan) in recognition of significant contributions in science and technology. He is Clarivate Analytics 2019 Highly Cited Researcher (Web of Science) and Foreign Fellow of EAJ.

Zhi Liu received his B.E. from the University of Science and Technology of China and his Ph.D. in informatics from the National Institute of Informatics, Japan. He is currently an Assistant Professor at Shizuoka University. He was a Junior Researcher (Assistant Professor) at Waseda University and a JSPS research fellow at the National Institute of Informatics. His research interests include video network transmission, vehicular networks, and mobile edge computing. He is a member of the IEEE and IEICE.

Hiroshi Masui received his Ph.D. in science from the Graduate School of Osaka University, Japan, in 1998. From 1998 to 2001, he was a researcher with Hokkaido University in Japan, and from 2002 to 2004, a JSPS researcher. Currently, he is a Professor at the School of Regional Innovation and Social Design Engineering and the Director of the Information Processing Center, Kitami Institute of Technology. His research interests include data science and distributed systems. He is a member of the IPS Japan.

Yusheng Ji received her B.E., M.E., and D.E. in electrical engineering from the University of Tokyo. She joined the National Center for Science Information Systems, Japan (NACSIS) in 1990. Currently, she is a Professor at the National Institute of Informatics (NII) and SOKENDAI (the Graduate University for Advanced Studies). Her research interests include network architecture, resource management, and quality-of-service provisioning in wired and wireless communication networks. She is/has been an Editor of the IEEE TVT, Associate Editor of IEICE Transactions and IPSJ Journal, Guest Editor-in-Chief, Guest Editor, and Guest Associate Editor of Special Issues of the IEICE Transactions and IPSJ Journal, and a Symposium Co-chair of IEEE GLOBECOM 2012, 2014, Track Chair of IEEE VTC 2016 Fall, 2017 Fall, and a TPC member of IEEE INFOCOM, ICC, GLOBECOM, WCNC, VTC, etc. She is IEEE Fellow from 2021.