Low-Carbon Geographically Distributed Cloud-Edge Task Scheduling

Yingjie Zhu[1(✉)], Ji Qi[2], Zehao Wang[1], Shengjie Wei[1], Yan Chen[1], Tuo Cao[1], Gangyi Luo[2], and Zhuzhong Qian[1(✉)]

[1] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
{yingjiezhu,zehaowang,mg20330069,dz21330047,tuocao}@smail.nju.edu.cn, qzz@nju.edu.cn

[2] China Mobile (Suzhou) Software Technology Co., Ltd., Suzhou, Jiangsu, China
{qiji,luogangyi2}@cmss.chinamobile.com

Abstract. Edge computing is a rapidly developing research area known for its ability to reduce latency and improve energy efficiency, and it also has a potential for green computing. Many geographically distributed edge servers are powered by renewable energy sources, due to the difficulties of using traditional power supplies or because of advancements in energy harvesting technologies. These green edge servers can cut down carbon emissions by processing tasks locally, but the inherent limitations of their computing capacity result in some tasks having to be uploaded to a data center to meet service-level agreement (SLA) requirements. To further reduce carbon emissions in cloud-edge systems, scheduling tasks to those low-carbon data centers while meeting latency constraints is highly beneficial. In this paper, we propose a low-carbon cloud-edge scheduling algorithm that utilizes Lyapunov optimization techniques and Markov approximation to address the long-term optimization problem of carbon emissions. Our algorithm guarantees provable performance, and simulation results demonstrate its effectiveness in striking a balance between carbon emissions and task latency.

Keywords: Low-carbon · Geo-distributed data centers · Edge-Cloud collaboration · Task scheduling

1 Introduction

Edge computing entails a decentralized computing architecture that strategically redistributes data storage and processing capabilities from centralized data centers to the network edge. By enabling on-premise execution of tasks on the edge server, it delivers low-latency and high-bandwidth services that facilitate real-time data analysis and processing. Furthermore, there are increasing scenarios

This work is funded by Nanjing University-China Mobile Communications Group Co., Ltd. Joint Institute.

T. Zhu et al. (Eds.): ICA3PP 2024, LNCS 15256, pp. 179–199, 2025.
https://doi.org/10.1007/978-981-96-1551-3_13

where edge servers are powered entirely by clean energy, which can be leveraged to achieve sustainable and environmentally friendly computing practices. For one thing, in many remote regions, it is difficult for edge servers to directly use wired power supplies due to the inconvenience of grid construction or the high cost of access to power supply [1]. For another, the geographically distributed nature of edge nodes allows for the utilization of on-site green energy with the aid of energy harvesting technologies [2,3]. Additionally, edge service providers are increasingly committed to building green edge servers in order to minimize their carbon footprint. A notable example is Akamai, which has pledged to power their edge servers with 100% renewable energy by 2030 [4]. These green edge servers emit significantly lower levels of carbon, thereby contributing to a reduction in carbon emissions when performing tasks on them.

However, the limited computing capabilities of edge servers pose a challenge when dealing with a large influx of tasks, especially those requiring significant computational resources. In such cases, the edge server may struggle to meet the service-level agreement (SLA) requirements due to prolonged processing times. Consequently, certain tasks have to be uploaded to cloud data centers equipped with ample computing resources, albeit at the expense of increased carbon emissions and the introduction of transmission latency.

Clean energy has been extensively integrated into the power infrastructure of data centers, and has been proven effective in reducing carbon emissions per unit of electricity consumption [5]. Nonetheless, the temporal and spatial distribution uncertainty of clean energy sources causes significant disparities in carbon emission rates among different data centers. Existing research on reducing carbon emissions of data centers mainly focuses on energy consumption reduction strategies, such as redirecting workloads to more energy-efficient data centers [6], dynamically turning off idle servers [7], scaling server speed during low workload periods [8], and using virtual machine placement (VMP) approaches [9]. However, these energy-saving measures do not necessarily result in carbon emission reduction for geographically distributed data centers, as some highly energy-efficient data centers may still exhibit high carbon emission rates due to the carbon intensity of the local energy supply. [10]. In this context, scheduling computational tasks to low-carbon data centers can effectively reduce total carbon emissions, but this strategy may potentially introduce additional transmission latency, as low-carbon data centers are often located farther away. Consequently, the scheduling of computational tasks presents a complex trade-off between the imperative to minimize carbon emissions and the requirement to maintain acceptable levels of latency.

This study focuses on a practical scenario that edge servers are interconnected with mulituple geo-distributed data centers. We introduce a task scheduling framework designed to harness the advantages of these distributed data centers and green edge servers, with the dual objectives of reducing overall carbon emissions and adhering to latency constraints. As illustrated in Fig. 1, computational tasks from users are first submitted to the edge servers, which then determine whether the tasks should be executed locally or uploaded to a specific data center for processing. We mathematically formulate the scheduling prob-

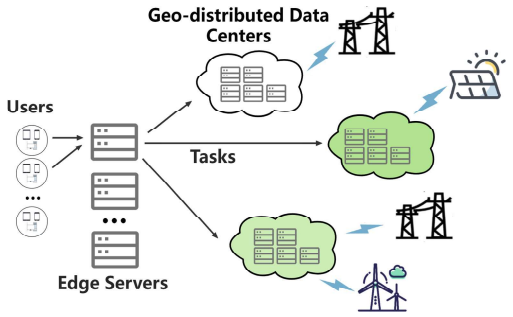

Fig. 1. An exemplification of the low-carbon geo-distributed cloud-edge task scheduling framework.

lem as a long-term optimization problem, aiming to minimize the cumulative carbon emissions, and the constraint is that the average processing time for each task remains within a predefined threshold, which reflects the system's SLA. To tackle this, we employ a Lyapunov-based online algorithm that decomposes the long-term problem into a series of single-time-slot sub-problems. Subsequently, we apply a Markov approximation-based algorithm to efficiently address each sub-problem within its respective time slot. Our proposed algorithm has theoretical performance guarantees, and we demonstrate the efficacy of our proposed algorithm through comprehensive simulations, highlighting its superior performance in reducing carbon emissions and task delay compared to alternative methodologies.

Our contributions in this work can be summarized as follows:

- We present a practical scheduling model of computational tasks in a scenario where edge servers are connected to geo-distributed data centers. By prioritizing carbon emission reduction as the primary goal rather than a constraint of the scheduling model, we significantly reduce the system's carbon emissions to the greatest extent possible. Instead of optimizing individual time slots, we mathematically formulate the scheduling problem as a long-term optimization objective to maximize the optimization of carbon emissions while ensuring the long-term stable operation of the system.
- We employ Lyapunov optimization techniques to derive a near-optimal policy that is independently of future information, and we design an efficient scheduling algorithm based on Markov approximation, which strikes a balance between acceptable time complexity and the preservation of solution effectiveness. Our methodologies are anchored by a robust theoretical framework, providing our proposed algorithm with a solid guarantee of theoretical soundness.
- We evaluate the performance of our algorithm through extensive simulations, and compare it with several benchmarks. These results suggest that our proposed algorithm has an advantage over other methods, and it is robust, adapt-

able, and effective in minimizing carbon emissions under a wide range of conditions.

The rest of this paper is structured as follows: Sect. 2 provides an overview of the related work. Section 3 presents the system model used in this study. Section 4 details the proposed algorithm. Section 5 showcases the results obtained from simulation experiments. Lastly, Sect. 6 concludes the paper.

2 Related Work

Some research has been conducted on reducing energy of edge computing system, but few studies have taken carbon emissions as the theme, and most of the efforts have focused on optimizing the cost at the edge by offloading strategies, ignoring the impact of data centers. Chen *et al.* [11] introduced an online peer offloading strategy that aims to optimize performance efficiency while considering long-term energy consumption constraints of edge servers. However, it is important to note that their peer offloading approach is specifically focused on interactions between edge servers. Xu *et al.* [1] used reinforcement learning to learn online offloading strategies, reducing long-term latency and operational costs in renewable energy-powered mobile edge systems, but they did not consider the impact of carbon emissions. In [12], Gu *et al.* proposed a virtual machine task migration strategy based on deep reinforcement learning to reduce carbon emissions in edge computing systems, but it only focuses on service tasks that can be migrated and deployed among edge servers, rather than computationally intensive tasks that consume more energy. The authors of [13] presented a resource-constrained randomized dependent rounding algorithm to enable offloading of machine learning tasks while operating within the constraints of limited carbon emission rights.

Geographical load balancing has also been explored to reduce system costs. However, most research efforts have focused on reducing energy consumption or electricity price. Liu *et al.* [14] distributed tasks to data centers with lower latency and electricity prices to reduce energy cost while ensuring latency requirements. Zhou *et al.* [10] experimentally demonstrated that saving energy cost due to spatial variations in carbon emission rates is not equivalent to reducing carbon emissions. They proposed a carbon emission-aware online control algorithm, COCA, based on Lyapunov optimization to minimize energy cost while satisfying carbon emission constraints. Lin *et al.* [15] further incorporated the uncertainty of on-site clean energy supply, but these works did not specifically model tasks and mainly considered request-type tasks with low computation and carbon emissions. Khosravi *et al.* [16] considered using a virtual machine placement strategy to reduce carbon emissions in multiple cloud data centers. However, it does not achieve flexible scheduling at the task level and primarily focuses on carbon pricing and carbon taxes, overlooking the temporal fluctuations in carbon emission rates of the power supply. The authors of [17] developed a renewable energy-aware multi-index job classification and scheduling methodology, which involves assigning workloads to data centers that possess an ample supply of renewable energy for efficient processing. However, these works only consider data center

task scheduling in the traditional cloud computing paradigm and are not applicable to the cloud-edge scenario addressed in this study.

Several prior studies have employed Lyapunov optimization algorithms to reduce system costs, such as energy and latency, in mobile edge systems. DREAM [18] used Lyapunov optimization algorithm to investigate the energy-delay tradeoff of mobile clouds that encompass diverse types of computation tasks. Mao *et al.* [19] integrated energy harvesting techniques into mobile-edge computing and proposed a Lyapunov-based algorithm to reduce the service cost in edge servers. Zhang *et al.* [20] used Lyapunov optimization to achieve the optimal arrangement of CPU-cycle frequencies for mobile devices and power levels for data transmission.

3 System Model

Within this section, we present an overview of our proposed system model of computational tasks in a geo-distributed cloud-edge system. Our model is based on the following components:

3.1 Cloud-Edge Scheduling Model

In the context of a network consisting of N interconnected edge servers and M data centers, we define various parameters to characterize the system. The computation capacity of the i-th edge server, denoted as $MIPS_i^E$, quantifies its ability to execute instructions per unit time. Similarly, the computation capacity of single server in the j-th data center is expressed as $MIPS_j^D$, and its power rate is P_j^D. The carbon emission rate of the j-th data center at time slot t is presented as $C_j^D(t)$, and its power usage effectiveness is denoted as PUE_j^D. Furthermore, the network encompasses transmission connections between each edge server and data center, necessitating consideration of parameters such as the physical hop count R_{ij} (corresponding to the distance), the power consumption per hop router for transmitting unit data P_{ij}^{net}, and the overall carbon emission rate of the link denoted as C_{ij}^{net}.

In each time slot t, the arrival rate of tasks in i-th edge server is $a_i(t)$, which means there will be $A_i(t)$ computational tasks, each characterized by distinct computation and data requirements. The computation amount of a task is quantified by the number of clock cycles it necessitates, denoted as $I_{(i,k)}(t)$. The data amount $D_{(i,k)}(t)$ represents the number of bits needed for transmitting the task. Binary variables $X_{(i,k)j}(t)$ are used to indicate whether the k-th computing task on the i-th edge server is uploaded to the j-th data center for processing during time slot t. If $X_{(i,k)j}(t)$ is set to 0 for all data centers j, it signifies that the task is scheduled to be executed locally on the edge server. Hence, we can

formulate the following constraints to govern the task scheduling:

$$\sum_{j=1}^{M} X_{(i,k)j}(t) \leq 1, \tag{1}$$

$$X_{(i,k)j}(t) \in \{0,1\}. \tag{2}$$

If the k-th task on the i-th edge server needs to be uploaded to the j-th data center, its bandwidth consumption can be denoted as $b_{(i,k)j}(t)$. All tasks uploaded from the i-th edge to the j-th data center share a common link with a total bandwidth limit denoted as $B_{ij}(t)$. To guarantee that the aggregate data volume of all transmitted tasks on this link remains within the available bandwidth, the following bandwidth constraint is formulated:

$$\sum_{k=1}^{A_i(t)} X_{(i,k)j}(t) \cdot b_{(i,k)j}(t) \leq B_{ij}(t). \tag{3}$$

3.2 Latency Model

The delay of the tasks on the i-th edge server, when processed locally, is predominantly influenced by the local computation time, which is represented as $T_i^E(t)$. As multiple tasks can be processed locally, they must share the computation capacity of the edge server. Consequently, the total delay of all tasks on the edge server can be derived using the following expression:

$$T_i^E(t) = \frac{\sum_{k=1}^{A_i(t)}(1 - \sum_{j=1}^{M} X_{(i,k)j}(t)) \times I_{(i,k)}(t)}{MIPS_i^E}. \tag{4}$$

When the k-th task on the i-th edge server is decided to be uploaded to the j-th data center, its delay can be denoted as $T_{(i,k)}^D(t)$ and the delay consists of two parts, transmission time and computing time:

$$T_{(i,k)}^D(t) = \sum_{j=1}^{N} X_{(i,k)j}(t) \times (TCal_{(i,k)j}(t) + TTrans_{(i,k)j}(t)). \tag{5}$$

The transmission time from the i-th edge node to the j-th data center depends on the data amount $D_{(i,k)}(t)$, physical hop count R_{ij}, and transmission bandwidth $b_{(i,k)j}(t)$, while the time for computing the task at the data center can be calculated by dividing computation amount by the server's computation capacity. Thus we can get two equations as follows:

$$TTrans_{(i,k)j}(t) = X_{(i,k)j}(t) \times R_{ij} \times \frac{D_{(i,k)}(t)}{b_{(i,k)j}(t)}, \tag{6}$$

$$TCal_{(i,k)j}(t) = X_{(i,k)j}(t) \times \frac{I_{(i,k)}(t)}{MIPS_j^D}. \tag{7}$$

Then we can get the average latency of all tasks in one slot:

$$T_A(t) = \frac{\sum_{i=1}^{N}(T_i^E(t) + \sum_{k=1}^{A_i(t)} T_{(i,k)j}^D(t))}{\sum_{i=1}^{N} A_i(t)}. \tag{8}$$

In order to meet the SLA requirements and sufficiently reduce long-term carbon emissions, we can tolerate a slightly larger average latency in a specific time slot, and simply require that the average delay across all time slots does not exceed the predifined value T_{avg}. Ultimately, the latency constraint can be formulated as follows:

$$\frac{1}{T}\sum_{t=1}^{T} T_A(t) \leq T_{avg}. \tag{9}$$

3.3 Carbon Emission Model

When the k-th task on the i-th edge server is selected to be uploaded to the j-th data center for processing, its carbon emissions comprise two main components: transmission carbon emissions $CETrans_{(i,k)j}(t)$ and computation carbon emissions $CECal_{(i,k)j}(t)$. Firstly, let's consider the computation carbon emissions generated by the data center's server when performing operations. The power consumption can be derived by multiplying the server's power rate P_j^D with computation time $TCal_{(i,k)j}(t) = \frac{I_{(i,k)}(t)}{MIPS_j^D}$. To obtain carbon emissions, we also need to multiply the energy utilization rate PUE_j^D and the carbon emission rate $C_j^D(t)$ of the data center, then we can get the equation as follows:

$$CECal_{(i,k)j}(t) = P_j^D \times \frac{I_{(i,k)}(t)}{MIPS_j^D} \times PUE_j \times C_j^D(t). \tag{10}$$

The transmission carbon emissions are determined by the data amount $D_{(i,k)}(t)$ and the number of physical hops R_{ij} that the transmission goes through. Similarly, we need to multiply the power consumption of each hop router to transmit unit data P_{ij}^{net} and the carbon emission rate C_{ij}^{net}. The formula is as follows:

$$CETrans_{(i,k)j}(t) = R_{ij} \times D_{(i,k)}(t) \times P_{ij}^{net} \times C_{ij}^{net}. \tag{11}$$

The sum of these two parts gives the carbon emissions generated by the k-th task when it is decided to be uploaded to the j-th data center for processing, so the total carbon emissions can be obtained by summing the carbon emissions of all tasks as follows:

$$SumCE(t) = \sum_{i=1}^{N}\sum_{k=1}^{A_i(t)}\left[\sum_{j=1}^{M} X_{(i,k)j}(t) \times (CECal_{(i,k)j}(t) + CETrans_{(i,k)j}(t))\right]. \tag{12}$$

3.4 Long-Term Optimization Problem

Our goal is to minimize the carbon emissions generated by the system while ensuring delay and bandwidth requirement, therefore the optimization problem can be formulated as:

$$\begin{aligned} P_1: \min \quad & \lim_{T\to+\infty} \frac{1}{T} \sum_{t=1}^{T} SumCE(t) \\ \text{s.t.} \quad & (1)-(3), (9). \end{aligned} \tag{13}$$

4 Algorithm Design

Due to the two major challenges in problem P_1: the long-term latency constraint and the NP-hardness of instantaneous decision-making, we employ an optimization algorithm design based on Lyapunov optimization [21] and Markov approximation. Firstly, we utilize Lyapunov functions to transform the long-term constraints and optimization objectives of the original problem into per-time-slot sub-problems. Then, we approximately solve the sub-problem in one time slot using the Markov approximation process.

4.1 Lyapunov-Based Online Algorithm

The optimization problem contains long-term optimization objectives and constraint inequalities, while the actual system does not know the situation of the subsequent time slots, and can only make decisions in the current time slot, which will lead to the potential violation of long-term constraints, or can not make full use of the elasticity space in a single time slot brought about by the long-term constraints, so it is necessary to firstly transform the original problem into per-time-slot decision sub-problems, which are then further solved.

To transform the original long-term optimization problem into per-time-slot sub-problems, we first define the Lyapunov virtual queue, denoted as $q(t)$, to represent the cumulative deviation between the delay at time slot t and the target constraint, with an initial value of 0 and a recursive equation of:

$$q(t+1) = max(0, T_A(t) - T_{avg} + q(t)). \tag{14}$$

A larger value of the queue $q(t)$ indicates a greater cumulative deviation of the delay from the original constraint up to time slot t. Therefore, our objective is to control the size of this queue. For this purpose we further define a Lyapunov function, $L(q(t)) = \frac{1}{2}(q(t))^2$, which measures the degree of congestion of the queue, and in order to make this function evolve to smaller states, we introduce a Lyapunov drift function within a single time slot to represent the growth of the queues' backlog from time slot t to time slot $(t+1)$:

$$\Delta(q(t)) = E[L(q(t+1)) - L(q(t)) \mid q(t)]. \tag{15}$$

Since $\Delta(q(t))$ represents the growth of the Lyapunov queue within a single time slot, we can ultimately restrict the queue size by minimizing $\Delta(q(t))$. This ensures compliance with the constraints of the original problem. To simultaneously minimize the objective while satisfying the constraints as much as possible, we can minimize both the objective and $\Delta(q(t))$ by introducing a parameter V to control the optimization weight. As a result, we obtain the Lyapunov drift-plus-penalty term as $\Delta(q(t)) + V \cdot \mathrm{SumCE}(t)$.

However, $\Delta(q(t))$ itself has a high computational complexity, and for this reason, we employ the Minimum Drift Plus Penalty algorithm from Lyapunov optimization theory, targeting the minimization of the upper bound on the drift plus penalty term, rather than directly minimizing the term itself. We first give the following lemma:

Lemma 1.

$$\Delta(q(t)) + V \cdot SumCE(t) \leq B + q(t) \cdot (T_A(t) - T_{avg}) + V \cdot SumCE(t), \quad (16)$$

where $B = \frac{1}{2}(T_{max} - T_{avg})^2$, $T_{max} = max_{t \in T} T_A(t)$ *represents the maximum worst-case delay within each time slot.*

Proof. From Eq. (15), we have:

$$\begin{aligned} \Delta(q(t)) &= \frac{1}{2} E[q(t+1)^2 - q(t)^2 \mid q(t)] \\ &\leq \frac{1}{2} E[(q(t) + T_A(t) - T_{avg})^2 - q(t)^2 \mid q(t)] \\ &= \frac{1}{2}(T_A(t) - T_{avg})^2 + q(t) E[(T_A(t) - T_{avg}) \mid q(t)] \\ &\leq \frac{1}{2}(\max_{t \in T} T_A(t) - T_{avg})^2 + q(t) \cdot (T_A(t) - T_{avg}). \end{aligned} \quad (17)$$

Remember $T_{max} = max_{t \in T} T_A(t)$ and $B = \frac{1}{2}(T_{max} - T_{avg})^2$, then we have:

$$\Delta(q(t)) + V \cdot SumCE(t) \leq B + q(t) \cdot (T_A(t) - T_{avg}) + V \cdot SumCE(t). \quad (18)$$

□

Therefore, we only need to minimize the expression $B + q(t) \cdot (T_A(t) - T_{avg}) + V \cdot \mathrm{SumCE}(t)$ to obtain an approximate solution. Since B is a constant, the transformed instantaneous decision problem is as follows:

$$\begin{aligned} P_2 : \min \quad & q(t) \cdot T_A(t) + V \cdot SumCE(t) \\ \text{s.t.} \quad & (1) - (3). \end{aligned} \quad (19)$$

Finally, we obtain an approximate sub-problem P_2 of the original problem that can be solved instantaneously, and we can obtain an approximate solution to the original problem by solving this problem. Additionally, we can control the trade-off between latency and the overall carbon emissions by adjusting the parameter V. Algorithm 1 presents our Lyapunov-based online algorithm framework, and we will solve problem P_2 in the next subsection.

Algorithm 1. Lyapunov-based Online algorithm

Input: T_{avg}, T;
$q(1) = 0$;
for $t = 1$ to T **do**
  Update system parameters;
  Use Algorithm to obtain scheduling strategy by solving P_2 ;
  $q(t+1) = max(0, T_A(t) - T_{avg} + q(t))$;
end for

4.2 Markov Approximation Based One-Slot Algorithm

For ease of expression, we denote $U(\alpha, \beta)$ as the objective function of P_2 where $\alpha = \{X_{(i,k)j}(t) \mid k \in \{1, ..., A_i(t)\}, j \in \{1, ..., M\}, i \in \{1, ..., N\}\}$ represents a task scheduling strategy, and $\beta = \{b_{(i,k)j}(t) \mid k \in \{1, ..., A_i(t)\}, j \in \{1, ..., M\}, i \in \{1, ..., N\}\}$ is the allocation of bandwidth. Notably, the constraint involving β is only reflected in the time constraint, and once α has been determined, each edge server can independently decide on the bandwidth allocation for its uploaded tasks. We can obtain β directly through the following theorem:

Theorem 1. *When α has been determined, the optimal choice for β to satisfy the bandwidth constraint is as follows:*

$$b_{(i,k)j}(t) = \frac{B_{ij} \cdot \sqrt{X_{(i,k)j}(t) \cdot D_{(i,k)}(t)}}{\sum_{k=1}^{A_i(t)} \sqrt{X_{(i,k)j}(t) \cdot D_{(i,k)}(t)}}. \tag{20}$$

Proof. It is straightforward to see that such a bandwidth allocation satisfies the bandwidth constraint, because:

$$\sum_{k=1}^{A_i(t)} X_{(i,k)j}(t) \cdot b_{(i,k)j}(t) = \frac{\sum_{k=1}^{A_i(t)} B_{ij} \cdot \sqrt{X_{(i,k)j}(t) \cdot D_{(i,k)}(t)}}{\sum_{k=1}^{A_i(t)} \sqrt{X_{(i,k)j}(t) \cdot D_{(i,k)}(t)}} = B_{ij}(t). \tag{21}$$

Secondly we prove its optimality. As shown in Sect. 3.2 on delay calculation, the bandwidth choice β only affects the transmission delay $TTrans_{(i,k)j}(t)$ of each task:

$$\begin{aligned}
\sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{A_i(t)} TTrans_{(i,k)j}(t) &= \sum_{i=1}^{N} \sum_{j=1}^{M} R_{ij} \times \sum_{k=1}^{A_i(t)} \frac{X_{(i,k)j}(t) \cdot D_{(i,k)}(t)}{b_{(i,k)j}(t)} \\
&\geq \sum_{i=1}^{N} \sum_{j=1}^{M} R_{ij} \times \frac{(\sum_{k=1}^{A_i(t)} \sqrt{X_{(i,k)j}(t) \cdot D_{(i,k)}(t)})^2}{\sum_{k=1}^{A_i(t)} b_{(i,k)j}(t)} \\
&\quad \text{(By using the Cauchy-Schwarz inequality)} \\
&\geq \sum_{i=1}^{N} \sum_{j=1}^{M} R_{ij} \times \frac{(\sum_{k=1}^{A_i(t)} \sqrt{X_{(i,k)j}(t) \cdot D_{(i,k)}(t)})^2}{B_{ij}(t)}.
\end{aligned} \tag{22}$$

In order to achieve its optimal value, β need to fulfill the condition of taking equality of two inequalities, so it can be calculated as:

$$b_{(i,k)j}(t) = \frac{B_{ij} \cdot \sqrt{X_{(i,k)j}(t) \cdot D_{(i,k)}(t)}}{\sum_{k=1}^{A_i(t)} \sqrt{X_{(i,k)j}(t) \cdot D_{(i,k)}(t)}}. \tag{23}$$

And this is the value of β. Therefore, β is both feasible and optimal. □

Thus, we are able to solve the optimization problem with single decision variable α. We will use the Markov approximation method to solve P_2 because it is still NP-hard for its combinatorial nature [22].

Algorithm 2. Markov Approximation Based One-slot Optimization Algorithm

Input: R_{max};
1: $\alpha = \{X_{(i,k)j}(t) = 0 \mid k \in \{1, ..., A_i(t)\}, j \in \{1, ..., M\}, i \in \{1, ..., N\}\}$;
2: Calculate β using Eq.(20);
3: $count \leftarrow 0$
4: **repeat**
5: $count \leftarrow count + 1$
6: Randomly change α to α';
7: **if** α' is feasible **then**
8: Calculate β' using Eq.(20);
9: Calculate $U(\alpha, \beta)$, $U(\alpha', \beta')$;
10: $\eta \leftarrow \frac{1}{1+e^{(U(\alpha',\beta')-U(\alpha,\beta))/r}}$;
11: with probability η replace α, β with α', β';
12: **end if**
13: **until** $count = R_{max}$ or there is no significant improvement for more than 10 iterations;
Output: α, β;

The online algorithm (described in Algorithm 2) applies Markov chain approximation [23] to continuously update the task scheduling policy α and obtain a near-optimal solution. The algorithm works in the following way: firstly, at the beginning of each time slot, we simply put all incoming computation task on edge servers by using assignment $\alpha = \{X_{(i,k)j}(t) = 0 \mid k \in \{1, ..., A_i(t)\}, j \in \{1, ..., M\}, i \in \{1, ..., N\}\}$. Secondly, we repeat the process as follows: we randomly change α to α' by selecting a task and changing its deployment (i.e., if it is scheduled to be uploaded to a data center, we can reschedule it on a different data center or reschedule it on the original edge server), then we can use Eq. (20) to obtain β', and further calculate the new objective value $U(\alpha', \beta')$ in problem P_2. calculate the probability of a state transition, denoted as η, based on the difference between objective values. In the current iteration, the new strategy replace the original one with probability η. Hence, the likelihood of changing the strategy is higher if the new strategy leads to a lower objective value. The parameter $r \geq 0$ (Line 10) is used to balance exploration and exploitation, a

bigger r will lead to a bigger probability η and thus encouraging accepting a new deployment, and when $r \to 0$, $\eta \to 0$, the algorithm will hardly accept new deployment. The iterative process is sustained until the criterion of R_{max} iterations is met or there is an absence of noteworthy enhancement in the objective value,(i.e., $|U(\alpha, \beta) - U(\alpha', \beta')| < 0.01$) for more than 10 iterations. As shown in [24], with the selection of appropriate parameters, the Markov approximation-based Algorithm is capable of achieving super-linear convergence.

4.3 Theoretic Analysis

Sine our algorithm uses a Lyapunov-based optimization framework and a Markov approximation based one-slot optimization algorithm, respectively, it has two main theoretical guarantees:

Theorem 2. *Our Lyapunov-based online algorithm implements the following performance constraints in terms of both the optimization objective of carbon emission rate and the constraint objective of queue stability:*

$$\lim_{T \to +\infty} \frac{\sum_{t=1}^{T} E[SumCE(t) \mid q(t)]}{T} \leq ce_{opt} + \frac{B}{V}, \tag{24}$$

$$\lim_{T \to +\infty} \frac{\sum_{t=1}^{T} E[q(t)]}{T} \leq \frac{B + V(ce_{max} - ce_{opt})}{\xi}. \tag{25}$$

where ce_{opt} is the theoretically optimal carbon emission value of the original problem P_1, ce_{max} is the highest carbon emission value among all feasible solutions of P_1, and $\xi > 0$ is a positive constant which will be used in the proof, specifically to indicate that there must exist a certain scheduling policy under the problem whose latency can always be less than the upper bound on latency in every time slot, and the cumulative delay constraint difference is ξ.

Proof. Due to space constraints, we give an abbreviated proof, and the detailed proof can be found in Theorem 4.8 of the [21]. Since the original problem P_1 is guaranteed to have a feasible solution, it also possesses an optimal solution and a vector of optimal scheduling strategy denoted as ce_{opt} and α_{opt}, respectively [25]. Therefore, we can state that:

$$\begin{aligned} &E[\Delta(q(t)) + V \cdot SumCE(t)|q(t)] \\ &\leq B + E[q(t) \cdot (T_A(\alpha_{opt}(t)) - T_{avg})] + V \cdot E[SumCE(\alpha_{opt}(t))] \\ &\leq B + V \cdot ce_{opt}. \end{aligned} \tag{26}$$

By summing up the derived conclusion over time slots, we obtain:

$$\begin{aligned}(B + V \cdot ce_{opt})T &\geq \sum_{t=1}^{T} E[\Delta(q(t)) + V \cdot SumCE(t)|q(t)] \\ &= E[L(q(T))] + V \cdot \sum_{t=1}^{T} E[SumCE(t)|q(t)] \\ &\geq V \cdot \sum_{t=1}^{T} E[SumCE(t)|q(t)].\end{aligned} \tag{27}$$

Thus,we can get our conclusion as:

$$\frac{1}{T}\sum_{t=1}^{T} E[SumCE(t)|q(t)] \leq ce_{opt} + \frac{B}{V}. \tag{28}$$

This establishes the first conclusion of Theorem 2. Next, we will prove the second conclusion. Assuming there is a vector of placement strategy α^* satisfies:

$$\exists \xi > 0, E[T_A(\alpha^*) - T_{avg}] \leq -\xi. \tag{29}$$

Since the carbon emissions objective in the original problem is bounded, with a lower bound of the optimal value ce_{opt} and an upper bound of ce_{max}, we can conclude that:

$$\Delta(q(t)) + V \cdot ce_{opt} \leq B + q(t) \cdot (T_A(\alpha^*) - T_{avg}) + V \cdot ce_{max}. \tag{30}$$

Taking the expectation of both sides of the above equation, we obtain:

$$\begin{aligned}E[\Delta(q(t))|q(t)] + V \cdot ce_{opt} &\leq B + E[q(t)]E[\cdot(T_A(\alpha^*) - T_{avg})] + V \cdot ce_{max} \\ &\leq B - \xi \cdot E[q(t)] + V \cdot ce_{max}.\end{aligned} \tag{31}$$

Thus, we can get:

$$E[L(q(t+1)) - L(q(t))] \leq B + V \cdot (ce_{max} - ce_{opt}) - \xi \cdot E[q(t)]. \tag{32}$$

By summing up the derived conclusion over time slots, we obtain:

$$E[L(q(T)) - L(q(1))] \leq T[B + V \cdot (ce_{max} - ce_{opt})] - \xi \sum_{t=0}^{T} \cdot E[q(t)]. \tag{33}$$

Thus, we can get:

$$\frac{1}{T}\sum_{t=1}^{T} \cdot E[q(t)] \leq \frac{B + V \cdot (ce_{max} - ce_{opt})}{\xi}. \tag{34}$$

□

From these two equivalences we can infer that there exists a trade-off between latency and carbon emissions is within $[O(1/V), O(V)]$, which means by changing the value of V we can control the performance of our algorithm. When we choose a larger control parameter V, optimization targeting carbon emissions yields better results, but the latency will also increase.

Theorem 3. *Our Markov approximation based one-slot optimization algorithm has the following performance guarantees:*

$$E[s_{local}(t)] \leq s_{opt}(t) + r \cdot \ln |\Omega| \tag{35}$$

where $s_{opt}(t)$ is the theoretical optimal solution of sub-problem P_2, and the online algorithm based on Markov approximation process proposed in this paper converges to the solution as $s_{local}(t)$, r is the system parameter regulating the state transfer probability in Algorithm, and Ω is the set of feasible solutions of P_2.

Proof. the detailed proof can be found in [26].

5 Performance Evaluation

In this section, we demonstrate the effectiveness of our algorithm by comparing it with several benchmarks in a simulation experiment based on real data. Besides, we verify the derived theoretical results mentioned above and discuss the impact of different parameters.

Table 1. Settings of experiment parameters

	parameters	value
Network device	transmission power(kW/Gbps)	1.28
	bandwidth(Mbps)	100
Data Center Server	calculate power rate(kW)	0.4
	main frequency(GHz)	8
	PUE	1.5
Edge Server	main frequency(GHz)	1.5
Carbon emission	thermal power(kgCO2/kW.h)	0.842

A. Simulation Setup

To ensure the reliability of our simulation results, we incorporate the clean energy ratio of data centers in different regions of China, which was provided by Greenpeace [27]. Our deployed data centers are strategically located in representative provinces such as Sichuan, Guizhou, Jiangsu, Guangdong and Shanghai, where

there are large differences in the carbon emission rates of data center power and transmission distances in these regions. Our simulation spans 576 time slots, each lasting 5 min. At the beginning of each time slot, the carbon emission rate for each data center is adjusted within a plausible range, following a normal distribution that is centered on the median. Concurrently, other system parameters remain constant and are detailed in Table 1. During each time slot, the number of newly arrived computing tasks of each edge node is uniformly distributed across different scales, and the task information, including computation and data amounts, is obtained from Intel Netbatch [28].

B. Performance Benchmark

In this section, we abbreviate our proposed algorithm as the Lyapunov Markov(LM) for convenience, as it is based on Lyapunov optimization and Markov approximation. We will compare its performance against the following benchmarks:

- **All-Cloud algorithm(ACloud)**: This algorithm adheres to the conventional cloud computing paradigm by disregarding the processing capabilities of edge servers and uploading all tasks from the edge to the nearest data center to achieve minimal latency.
- **Random Load Balance(RLB)**: This algorithm, while ensuring latency requirements are met, initially selects a subset of tasks for execution at the edge, and then uploads a portion of the remaining tasks to randomly selected distant low-carbon data centers, rather than uploading all to the nearest data center. This approach not only maintains a stable load across various data centers but also effectively reduces carbon emissions.
- **Delay-optimal Markov(DoM)**: This single-time-slot algorithm simply optimizes for latency as the primary objective. By implementing a Markov approximation process algorithm with the same parameters as in the LM algorithm, it achieves a cloud-edge scheduling algorithm with minimal latency within a single time slot, without significantly deviating from the carbon emission constraints.
- **Carbon-optimal Markov(CoM)**: This single-time-slot algorithm simply optimizes for carbon emissions as the primary objective. By implementing a Markov approximation process algorithm with the same parameters as in the LM algorithm, it achieves a cloud-edge scheduling algorithm with minimal carbon emissions within a single time slot, without significantly deviating from the latency constraints.

In summary, ACloud simulates a traditional cloud computing scheduling algorithm that prioritizes latency, while RLB emulates a heuristic geo-distributed cloud-edge optimization strategy. DoM and CoM, which both employ a singular optimization goal within a Markov approximation framework with identical parameters, are specifically designed to underscore the comparative advantages of the Lyapunov optimization approach in achieving a balance between latency and carbon emissions.

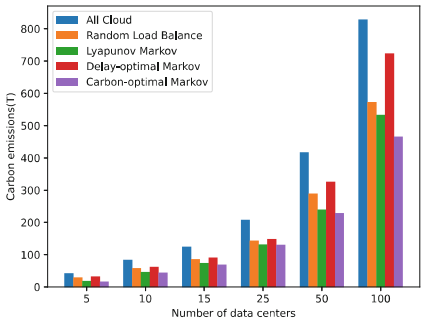

Fig. 2. Carbon emissions of five algorithms under different system configurations.

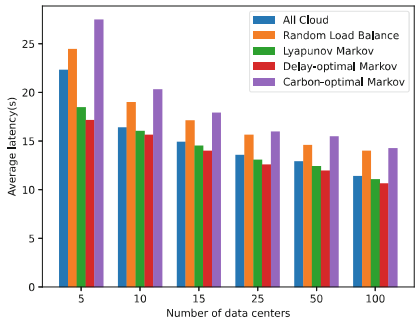

Fig. 3. Average system delay of five algorithms under different system configurations.

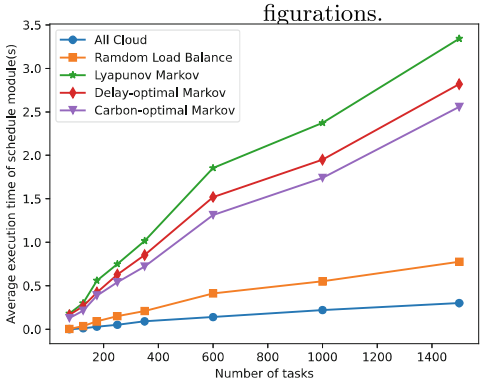

Fig. 4. Average execution time of five algorithms under different loading conditions.

C. Algorithm Comparison

Figure 2 and Fig. 3 depict the carbon emissions and average latency of five algorithms under different system configurations. For each configuration with a fixed number of data centers, we conducted tests across various task scales and integrated the results through a weighted ensemble approach, ultimately deriving a comprehensive assessment of system carbon emissions and latency for different configurations of data center counts.

From the figures, we observe that focusing exclusively on latency with greedy scheduling and failing to utilize the edge capabilities, ACloud results in the worst carbon emissions and also exhibits a lower degree of optimization in terms of latency. RLB reduces carbon emissions through load balancing between edge nodes and geo-distributed data centers, but at the cost of higher latency due to its random scheduling approach. DoM and CoM represent two extremes in the trade-off between carbon emissions and latency. Although DoM achieves the lowest average latency, it incurs significant carbon emissions, deviating from our original design objective. Similarly, CoM minimizes carbon emissions but causes intolerable latency. Our algorithm, on the contrary, achieves a large optimization

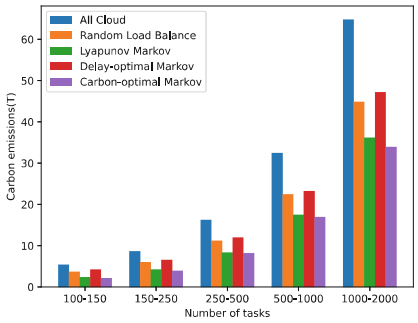

Fig. 5. Carbon emissions of five algorithms under different loading conditions.

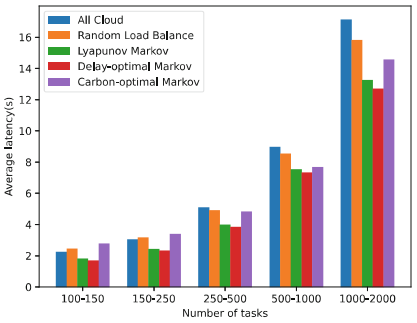

Fig. 6. Average system delay of five algorithms under different loading conditions.

in both the delay and carbon emission dimensions, both of which are close to the optimal value, and it demonstrates stability even in large-scale scenarios.

Figure 4 illustrates the average execution time of five algorithms for each time slot under different loading conditions. Although algorithms based on Markov approximation have relatively slower execution times compared to ACloud and RLB, considering that a time slot is 5 min long, the execution time of few seconds is still acceptable. In practice, system administrator can easily reduce the execution time of the scheduling module by appropriately decreasing the values of the system parameters r and R_{max}, which will be further discussed below.

Figures 5 and 6 showcase the carbon emissions and average latency of five algorithms under varying loading conditions when $M = 5$. As the workload intensifies, the number of tasks that can be efficiently executed on the edge servers tends to reach a stable state. Consequently, both the overall carbon emissions and latency experience an upward trend. Nevertheless, our algorithm maintains its superiority throughout this process, primarily attributed to its advanced cross-regional data center scheduling capability.

D. The Impact of Different Parameters

1) **Convergence**: Figure 7 shows the process of converging to the minimum value of the optimization objective with the increase of the number of iterations, where the parameters of the different curves are the system parameter r set in the Algorithm, which affects the probability η of changing the state in the Markov approximation algorithm. The larger r is, the larger the probability is, which indicates that the state of the system is more easily to be changed, and the algorithm will have a larger space for exploring, but it will be equally more difficult to converge to a fixed value, and sometimes even miss the minimum value. However, if r is too small, the state of the system is difficult to be changed, and it may be difficult to find the optimal solution. From Fig. 7, it can be found that the algorithm works best when $r = 0.1$, which indicates that the selection should be moderate, and not too large or too small.

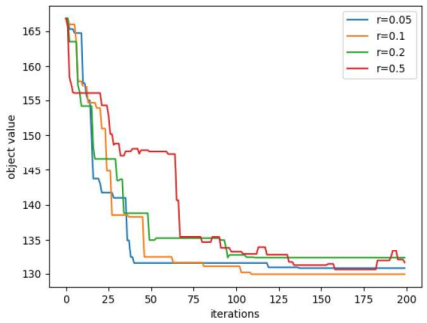

Fig. 7. The impact of different values of r in Algorithm.

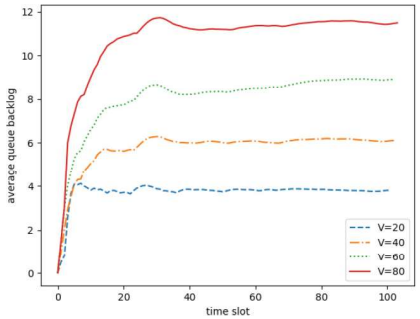

Fig. 8. The impact of different values of V.

2) **Weight-parameter**: The parameter V controls the trade-off between carbon emissions and latency in the optimization objective of the sub-problem within a single time slot, and as V increases this trade-off is biased in favour of carbon emissions and leads to larger and larger virtual queue backlog controlling the delay constraints, which can be seen in Fig. 8. We observe that the average queue size gradually stabilises as the time slot represented by the x-axis increases, suggesting that the Lyapunov optimization does indeed provide an effective constraint on the task latency, while the positive correlation between the queue size and the value of V suggests a controlling effect of the parameter V.
3) **Iteration-parameter**: In Fig. 9 we can find that the system parameter T_{avg} also directly controls the trade-off between latency and carbon emissions, when T_{avg} is small it represents a tighter constraint on delay and it is difficult to optimize the carbon emissions, while a larger T_{avg} indicates that the constraints on delay are looser and the carbon emissions can be optimized to a greater extent. While R_{max} is the iteration number parameter, it is obvious that the approximate solution obtained by the system will gradually converge to the theoretical optimal solution when the number of iterations is higher, but it will lead to more algorithm execution time. The degree of closeness of the optimal solution and the algorithm execution time are also a pair of trade-offs.

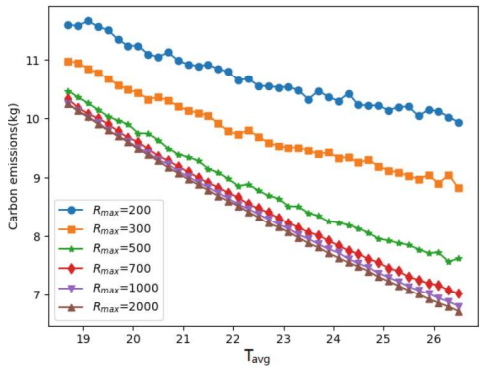

Fig. 9. The impact of different values of T_{avg} and R_{max}.

6 Conclusion

In this paper we study the overall carbon emissions and latency of geographically distributed cloud-edge systems. We propose an online scheduling algorithm based on Lyapunov optimization techniques and Markov approximation to decouple the long-term optimization problem into individual time slots and optimize the trade-off between carbon emissions and latency. We prove the performance guarantees of our proposed algorithm on a theoretical level and compare it with other algorithms through simulation experiments to demonstrate the superiority of our proposed algorithm.

References

1. Xu, J., Ren, S.: Online learning for offloading and autoscaling in renewable-powered mobile edge computing. In: 2016 IEEE Global Communications Conference (GLOBECOM), pp. 1–6 (2016). https://doi.org/10.1109/GLOCOM.2016.7842069
2. Han, T., Ansari, N.: A traffic load balancing framework for software-defined radio access networks powered by hybrid energy sources. IEEE/ACM Trans. Networking **24**(2), 1038–1051 (2016). https://doi.org/10.1109/TNET.2015.2404576
3. Li, W., et al.: On enabling sustainable edge computing with renewable energy resources. IEEE Commun. Mag. **56**(5), 94–101 (2018). https://doi.org/10.1109/MCOM.2018.1700888
4. Akamai: Akamai sustainability report 2021 (2021). https://www.akamai.com/zh/resources/research-paper/akamai-sustainability-report-2021
5. Lemay, M., Nguyen, K.K., St. Arnaud, B., Cheriet, M.: Toward a zero-carbon network: converging cloud computing and network virtualization. IEEE Internet Comput. **16**(6), 51–59 (2012). https://doi.org/10.1109/MIC.2011.128
6. Xu, H., Feng, C., Li, B.: Temperature aware workload managementin geo-distributed data centers. IEEE Trans. Parallel Distrib. Syst. **26**(6), 1743–1753 (2015). https://doi.org/10.1109/TPDS.2014.2325836

7. Lin, M., Wierman, A., Andrew, L.L.H., Thereska, E.: Dynamic right-sizing for power-proportional data centers. In: 2011 Proceedings IEEE INFOCOM, pp. 1098–1106 (2011). https://doi.org/10.1109/INFCOM.2011.5934885
8. Wierman, A., Andrew, L.L.H., Tang, A.: Power-aware speed scaling in processor sharing systems. In: IEEE INFOCOM 2009, pp. 2007–2015 (2009). https://doi.org/10.1109/INFCOM.2009.5062123
9. Ibrahim, A., Noshy, M., Ali, H.A., Badawy, M.: Papso: a power-aware VM placement technique based on particle swarm optimization. IEEE Access **8**, 81747–81764 (2020). https://doi.org/10.1109/ACCESS.2020.2990828
10. Zhou, Z., Liu, F., Zou, R., Liu, J., Xu, H., Jin, H.: Carbon-aware online control of geo-distributed cloud services. IEEE Trans. Parallel Distrib. Syst. **27**(9), 2506–2519 (2016). https://doi.org/10.1109/TPDS.2015.2504978
11. Chen, L., Zhou, S., Xu, J.: Computation peer offloading for energy-constrained mobile edge computing in small-cell networks. IEEE/ACM Trans. Networking **26**(4), 1619–1632 (2018). https://doi.org/10.1109/TNET.2018.2841758
12. Gu, L., Zhang, W., Wang, Z., Zeng, D., Jin, H.: Service management and energy scheduling toward low-carbon edge computing. IEEE Trans. Sustain. Comput. **8**(1), 109–119 (2023). https://doi.org/10.1109/TSUSC.2022.3210564
13. Ma, H., Zhou, Z., Zhang, X., Chen, X.: Toward carbon-neutral edge computing: greening edge AI by harnessing spot and future carbon markets. IEEE Internet Things J. **10**(18), 16637–16649 (2023). https://doi.org/10.1109/JIOT.2023.3268339
14. Liu, Z., Lin, M., Wierman, A., Low, S., Andrew, L.L.H.: Greening geographical load balancing. IEEE/ACM Trans. Networking **23**(2), 657–671 (2015). https://doi.org/10.1109/TNET.2014.2308295
15. Lin, W.T., Chen, G., Li, H.: Carbon-aware load balance control of data centers with renewable generations. IEEE Trans. Cloud Comput. **11**(2), 1111–1121 (2023). https://doi.org/10.1109/TCC.2022.3150391
16. Khosravi, A., Andrew, L.L.H., Buyya, R.: Dynamic VM placement method for minimizing energy and carbon cost in geographically distributed cloud data centers. IEEE Trans. Sustain. Comput. **2**(2), 183–196 (2017). https://doi.org/10.1109/TSUSC.2017.2709980
17. Kumar, N., Aujla, G.S., Garg, S., Kaur, K., Ranjan, R., Garg, S.K.: Renewable energy-based multi-indexed job classification and container management scheme for sustainability of cloud data centers. IEEE Trans. Industr. Inf. **15**(5), 2947–2957 (2019). https://doi.org/10.1109/TII.2018.2800693
18. Kwak, J., Kim, Y., Lee, J., Chong, S.: Dream: dynamic resource and task allocation for energy minimization in mobile cloud systems. IEEE J. Sel. Areas Commun. **33**(12), 2510–2523 (2015). https://doi.org/10.1109/JSAC.2015.2478718
19. Mao, Y., Zhang, J., Letaief, K.B.: Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE J. Sel. Areas Commun. **34**(12), 3590–3605 (2016). https://doi.org/10.1109/JSAC.2016.2611964
20. Zhang, G., Zhang, W., Cao, Y., Li, D., Wang, L.: Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices. IEEE Trans. Industr. Inf. **14**(10), 4642–4655 (2018). https://doi.org/10.1109/TII.2018.2843365
21. Neely, M.: Stochastic network optimization with application to communication and queueing systems. Springer (2010)
22. Xu, J., Chen, L., Zhou, P.: Joint service caching and task offloading for mobile edge computing in dense networks. In: IEEE INFOCOM 2018 - IEEE Confer-

ence on Computer Communications, pp. 207–215 (2018).https://doi.org/10.1109/INFOCOM.2018.8485977
23. Liu, F., Shu, P., Lui, J.C.: Appatp: an energy conserving adaptive mobile-cloud transmission protocol. IEEE Trans. Comput. **64**(11), 3051–3063 (2015). https://doi.org/10.1109/TC.2015.2401032
24. Chen, M., Liew, S.C., Shao, Z., Kai, C.: Markov approximation for combinatorial network optimization. IEEE Trans. Inf. Theory **59**(10), 6301–6327 (2013). https://doi.org/10.1109/TIT.2013.2268923
25. Ouyang, T., Zhou, Z., Chen, X.: Follow me at the edge: mobility-aware dynamic service placement for mobile edge computing. IEEE J. Sel. Areas Commun. **36**(10), 2333–2345 (2018). https://doi.org/10.1109/JSAC.2018.2869954
26. Cao, T., Qian, Z., Wu, K., Zhou, M., Jin, Y.: Service placement and bandwidth allocation for MEC-enabled mobile cloud gaming. In: 2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 179–188 (2021). https://doi.org/10.1109/WoWMoM51794.2021.00031
27. Greenpeace: Lighting up the green cloud: A study of data center energy consumption and renewable energy potential in China (2019). https://www.greenpeace.org.cn/wp-content/uploads/2019/09/
28. Shai, O., Shmueli, E., Feitelson, D.G.: Heuristics for resource matching in intel's compute farm. In: Desai, N., Cirne, W. (eds.) Job Scheduling Strategies for Parallel Processing, pp. 116–135. Springer, Heidelberg (2014)