Dynamic Carbon-Aware Hybrid Task Offloading for Collaborative Edge-Cloud Computing

Pin-Chun Chen, Li-Hsing Yen, Yan-Wei Chen, Ze-Yu Jin, and Chien-Chao Tseng
Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan.

***Abstract*—Collaborative edge-cloud computing is an emerging paradigm that balances processing efficiency and user experience by offloading edge workloads to the cloud (i.e., vertical offloading). Adding the option of horizontal offloading (i.e., edge-to-edge offloading) can further enhance the flexibility of this architecture. However, prior work ignored temporal and spatial variations in carbon emissions when making offloading decisions. Our study aims to jointly optimize the carbon footprint and the balance of edge servers' workloads, which is challenged by dynamic workload arrivals to edge servers and time-and-location-varying carbon intensity (CI). We propose an online algorithm Lyapunov optimization framework to configure offloading workload and transmission power for mixed vertical and horizontal offloading. Simulations based on historical CI data reveal that the proposed algorithm can efficiently reduce carbon footprint while balancing queue backlogs among edge and cloud servers.**

I. Introduction

Cloud computing offers substantial computing power but may not meet the stringent latency requirements of many emerging Internet of Things (IoT) applications [1]. By contrast, edge computing brings computing resources closer to end users, which can potentially reduce latency and enhance the quality of service for delay-sensitive applications.

Both cloud and edge environments introduce a new set of concerns, particularly in environmental sustainability [2]. The Information and Communication Technology (ICT) sector, including edge computing infrastructure, has been recognized as a significant contributor to global carbon emissions [3]. A prior study reported that the ICT industry could account for up to 14% of global carbon emissions by 2040 if current trends continue unchecked [4]. This growing environmental impact has sparked a renewed interest in developing green computing solutions [5]. However, most existing research on carbon-aware computing has focused primarily on cloud data centers, with limited attention given to the unique challenges posed by edge computing environments [6]. The distributed nature of edge systems, coupled with the heterogeneity of edge servers and the dynamic nature of IoT workloads, necessitates a more nuanced approach to carbon-aware computing.

This paper considers a collaborative edge-cloud system where geo-distributed edge servers are connected to a central cloud (Fig. 1). Because edge servers have constrained computing power and resources, we may perform a *vertical offloading* which strategically delegates computing tasks from edge servers to the cloud. We also consider *horizontal* or *peer* offloading, which delegates workload between edge servers. Specifically, each edge server takes individual workloads from IoT devices, sets up computing power to control the volume of workload processed locally, and adjusts transmission power and workload volume for horizontal and vertical offloading. The cloud server also performs a computing power control for each offloaded task. The goal is to balance edge and cloud servers' workloads (measured by server queue backlogs) while minimizing overall carbon footprint, which is challenged by the stochastic nature of workload arrival and time- and location-dependent carbon intensity (CI) defined as the amount of carbon emission when generating 1 kWh electricity (Fig. 2). To make it manageable, we apply the Lyapunov optimization framework to reformulate it as a stepwise deterministic problem. The proposed Dynamic Workload Power Allocation (DWPA) algorithm effectively reduces long-term carbon emissions while simplifying the problem-solving process, making it feasible and efficient for practical applications. We conducted simulations using historical CI dataset [7] for performance comparison between DWPA and other alternatives. The results confirm DWPA's ability to lower its carbon footprint while

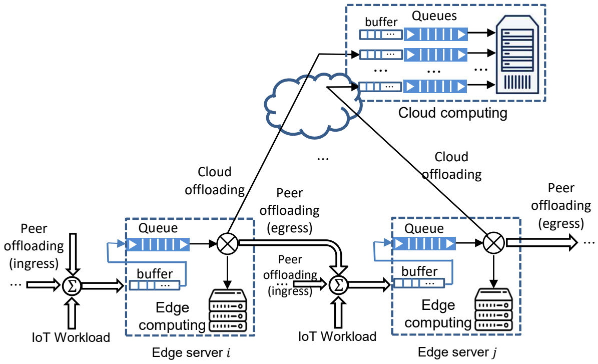


Figure 1: A collaborative edge-cloud system

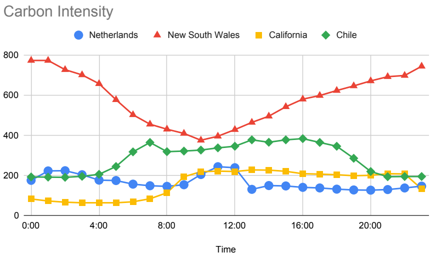


Figure 2: Typical daily changes of carbon intensity (unit: gCO2eq/kWh) in four different cities (Source: [7]).

maintaining stable queue backlogs.

The remainder of this paper is organized as follows: Sec. II reviews related work. Sec. III presents workload offloading and carbon footprint models. Sec. IV formulates the problem, and Sec. V details the proposed DWPA algorithm. The following section presents our simulation results and performance analysis; the last section concludes the paper.

II. Related Work

The granularity of task offloading can be binary or fractional. The former is an all-or-nothing decision while the latter allows for dividing tasks into subtasks, some of which may be locally processed while others are offloaded outwards [8]. Existing research efforts on offloading in collaborative edge–cloud environments have pursued different objectives. However, few studies addressed both carbon footprint and queue stability. For binary offloading, some early works considered energy cost but not queue stability [9, 10]. Recently, Chen et al. [11] proposed an online binary offloading algorithm to minimize energy consumption while maintaining queue stability. Some fractional-offloading studies considered queue stability but neither energy nor carbon cost [12]. Mao et al. [8] proposed a Lyapunov optimization framework to make a trade-off between energy consumption and queue backlog.

All the studies mentioned above considered energy consumption rather than carbon emission. Although higher energy consumption generally implies greater carbon emission, reducing energy consumption alone may not lead to carbon reduction [13] due to time- and location-dependent CI values. A fine-grained approach to carbon reduction is to consider CI directly. Chen et al. [14] proposed an online joint energy-network resource scheduling algorithm with an aim to minimize long-term average operational expenditure (OPEX) due to carbon trading. However, they assumed constant CI values. Yang et al. [15] considered three-tier vertical offloading and proposed an online algorithm to reduce carbon emissions while maintaining stable queue lengths. However, their work considered vertical offloading only and assumed device-dependent CI without considering the temporal variation of CI. By contrast, we considered mixed vertical-horizontal offloading and location- and time-dependent CI. Embracing horizontal offloading enhances flexibility but also adds another dimension to problem-solving: besides determining the workload volume to offload, we also need to choose the direction of offloading.

III. System Model

A. *Workload Offloading Model*

We consider a system consisting of a cloud server and a set of s edge servers $S = \{M_1, M_2, \cdots, M_s\}$. These servers work collaboratively with operation time divided into fixed-length time slots. The first part of each time slot with length T^{off} is allocated for transferring workloads while the rest with length T^{cmp} is for computation. In each time slot t, every edge server M_i receives IoT workload with volume $A_i(t)$ and keeps it in a local buffer. Besides the IoT workload, the buffer also keeps all the inbound offloading workload in time slot t. The contents of the buffer will be moved to a *task backlog queue* for processing in the next time slot. In each time slot, M_i can process some queued workload locally, offload some to other edge servers or the cloud, and leave some in the queue. Let $EC_i(t)$, $x_{i,j}(t)$, $x_i^{\text{cloud}}(t)$, be the volumes (in bits) of workload locally processed by M_i, offloaded from M_i to M_j, and offloaded from M_i to the cloud, respectively, in time slot t. Let $Q_i(t)$ denote the length (in bits) of M_i's queue at the beginning of time t. The following equation captures how the queue length evolves:

$$Q_i(t+1) = [Q_i(t) - EC_i(t) - x_i(t)]^+ + A_i(t) + \sum_{j \neq i} x_{j,i}(t), \quad (1)$$

where $x_i(t) = x_i^{\text{cloud}}(t) + \sum_{j \neq i} x_{i,j}(t)$ is the total volume of M_i's workload offloaded outwards in time slot t.

Our approach controls $EC_i(t)$ for each edge server M_i by setting up its CPU-cycle frequency in time slot t, $f_i(t)$, as

$$EC_i(t) = \frac{f_i(t) T^{\text{cmp}}}{\phi_i}, \quad (2)$$

where ϕ_i is M_i's computational intensity (CPU cycles required per bit of workload).

We assume that edge servers use wireless communications for workload transfer. (If this is not the case, we can omit carbon footprint due to wireless transmissions.) We control x_i for each M_i by setting up its transmission power and, thus, the transmission rate. Let $p_{i,j}(t)$ and $p_i^{\text{cloud}}(t)$ be M_i's transmission power for the peer offloading from M_i to M_j and the vertical offloading to the cloud, respectively. Then $R_{i,j}(t)$ and $R_{i,c}(t)$, which are the respective transmission rates (in bits per second), are

$$\begin{aligned} R_{i,j}(t) &= W_i \log_2 \left(1 + g_{i,j}(t) p_{i,j}(t)/N_0\right) \text{ and} \\ R_{i,c}(t) &= W_i \log_2 \left(1 + g_{i,c}(t) p_i^{\text{cloud}}(t)/N_0\right), \end{aligned} \quad (3)$$

where W_i is the bandwidth allocated to M_i, $g_{i,j}(t)$ and $g_{i,c}(t)$ are respective channel gains, and N_0 is the background noise power.

The cloud server has a buffer and a task queue q_i^{cloud} for workload offloaded from each edge server M_i. Let $Q_i^{\text{cloud}}(t)$ be the length of q_i^{cloud} in time slot t. We have the following equation for it:

$$Q_i^{\text{cloud}}(t+1) = [Q_i^{\text{cloud}}(t) - CC_i(t)]^+ + x_i^{\text{cloud}}(t), \quad (4)$$

where $CC_i(t)$ is the volume of workload in q_i^{cloud} that is processed by the cloud server in time slot t. Our approach controls $CC_i(t)$ by adjusting the cloud server's CPU-cycle frequency in time slot t, $f_i^{\text{cloud}}(t)$, as

$$CC_i(t) = \frac{f_i^{\text{cloud}}(t) T^{\text{cmp}}}{\phi^{\text{cloud}}}, \quad (5)$$

where ϕ^{cloud} is the cloud server's computational intensity (CPU cycles required per bit of workload).

B. Carbon Footprint Model

Since our actions does not affect embodied carbon emission, we ignore it and focus on operational carbon emissions. This perspective is consistent with standard practice for calculating carbon emissions from systems (e.g., [15]).

Let $k_i(t)$ be the CI for edge server M_i in time slot t. The amount of carbon emission for M_i's local processing in time slot t is $U_i(t) := k_i(t)\xi_i f_i(t)^3 T^{\text{cmp}}$, where ξ_i is M_i's CPU effective capacitance coefficient. Similarly, the cloud server's carbon footprint for processing workload offloaded from M_i in time slot t is $U_i^{\text{cloud}}(t) = k^{\text{cloud}}(t)\xi^{\text{cloud}} f_i^{\text{cloud}}(t)^3 T^{\text{cmp}}$, where $k^{\text{cloud}}(t)$ is the CI for the cloud server's power supply in time slot t and ξ^{cloud} is the cloud server's CPU effective capacitance coefficient.

Let $T_{i,j}^{\text{off}} = x_{i,j}(t)/R_{i,j}(t)$ and $T_{i,c}^{\text{off}} = x_i^{\text{cloud}}(t)/R_{i,c}(t)$ be the lengths of time for M_i offloading workloads to M_j and to the cloud, respectively. The amount of carbon emission for workload transmission from M_i to M_j is $U_{i,j}^{\text{off}}(t) = k_i(t)p_{i,j}(t)T_{i,j}^{\text{off}}$ while that for M_i's vertical offloading is $U_{i,c}^{\text{off}}(t) = k_i(t)p_i^{\text{cloud}}(t)T_{i,c}^{\text{off}}$. Therefore, the total carbon footprint of the system in time slot t is

$$o(t) = \sum_{i=1}^{s} \big[U_i(t) + \sum_{j\neq i} U_{i,j}^{\text{off}}(t) + U_{i,c}^{\text{off}}(t) + U_i^{\text{cloud}}(t)\big]. \quad (6)$$

IV. Problem Definition

At the beginning of each time slot, we determine each edge server's and cloud server's CPU-cycle frequency and transmission power in this slot. Our goal is twofold: maintaining queue stability and minimizing long-term carbon footprint. The following two subsections address these two goals, respectively. The last subsection presents a unified goal.

A. Maintaining Queue Stability

For the goal of maintaining queue stability, we employ the Lyapunov optimization framework, which has proven particularly effective for ensuring queue stability and long-term performance optimization in vertical offloading [8, 12, 11, 15]. Let $\Phi(t) = \{Q_i(t), Q_i^{\text{cloud}}(t)\}_{i\in S}$ be the states of all the edge and cloud server queues. We define a Lyapunov function based on $\Phi(t)$ to quantify the system's congestion level:

$$L(\Phi(t)) := \frac{1}{2}\sum_{i=1}^{s} \big[Q_i(t)^2 + Q_i^{\text{cloud}}(t)^2\big]. \quad (7)$$

$L(\cdot)$ is non-negative, and its value is zero only when all queues are empty. We then define the Lyapunov drift function as one-step conditional drift of $L(\cdot)$:

$$\Delta(\Phi(t)) := \mathbb{E}\{L(\Phi(t+1)) - L(\Phi(t)) \mid \Phi(t)\}. \quad (8)$$

If $\text{EC}_i(t)$, $\text{CC}_i(t)$, $x_{i,j}(t)$, $x_i^{\text{cloud}}(t)$ and $A_i(t)$ are all upper-bounded, we can show that $\Delta(\Phi(t))$ is also upper-bounded (we omit the proof due to space limitation):

$$\begin{aligned}
&\Delta(\Phi(t)) \le B \\
&+ \sum_{i=1}^{s} \{Q_i(t)\big[A_i(t) + \sum_{j\neq i} x_{j,i}(t) - \text{EC}_i(t) - x_i(t)\big] \mid \Phi(t)\} \quad (9) \\
&+ \sum_{i=1}^{s} \big\{Q_i^{\text{cloud}}(t)\big[x_i^{\text{cloud}}(t) - \text{CC}_i(t)\big] \mid \Phi(t)\big\}, \quad (10)
\end{aligned}$$

where B is a finite constant. Therefore, we can ensure queue stability by making $A_i(t) + \sum_{j\neq i} x_{j,i}(t) - \text{EC}_i(t) - x_i(t)$ in (9) and $x_i^{\text{cloud}}(t) - \text{CC}_i(t)$ in (10) non-positive on average.

B. Minimizing Carbon Footprint

The problem of minimizing long-term carbon footprint falls into the Mixed Integer Non-Linear Programming (MINLP) category, which has been demonstrated to be NP-hard. Moreover, the stochastic nature of CI and the potential for multiple local optima make traditional optimization methods unsuited. Most importantly, minimizing long-term carbon footprint and maintaining queue stability are disparate objectives.

The Lyapunov optimization framework allows us to unify the seemingly disparate objectives. We define a cost function $C(t)$ to unify the drift and the carbon footprint in time slot t:

$$C(t) := \Delta(\Phi(t)) + V\mathbb{E}\{o(t) \mid \Phi(t)\}, \quad (11)$$

where V controls the optimization priority between the two key performance indicators. The goal is

$$\text{P} : \min_{\Gamma(t)} C(t), \quad (12)$$

where $\Gamma(t) = \{f_i(t), f_i^{\text{cloud}}(t), p_{i,j}(t), p_i^{\text{cloud}}(t), x_{i,j}(t), x_{i,c}(t)\}$ is the set of decision variables. P is subject to the following constraints:

$$\begin{aligned}
&\text{C1} : 0 \le f_i(t) \le f_i^{\max},\ \forall i \\
&\text{C2} : 0 \le f_i^{\text{cloud}}(t) \le f_i^{\text{cmax}},\ \forall i \\
&\text{C3} : 0 \le p_{i,j}(t) \le p_{i,j}^{\max},\ \forall i \\
&\text{C4} : 0 \le p_i^{\text{cloud}}(t) \le p_i^{\text{cmax}},\ \forall i \\
&\text{C5} : EC_i(t) + x_i^{\text{cloud}}(t) + \sum_{j\neq i}^{s} x_{i,j}(t) \le Q_i(t),\ \forall i \quad (13) \\
&\text{C6} : CC_i(t) \le Q_i^{\text{cloud}}(t),\ \forall i \\
&\text{C7} : \lim_{t\to\infty} \frac{\mathbb{E}[Q_i(t)]}{t} = 0, \forall i \\
&\text{C8} : \lim_{t\to\infty} \frac{\mathbb{E}[Q_i^{\text{cloud}}(t)]}{t} = 0, \forall i.
\end{aligned}$$

Constraints C1 and C2, respectively, specify the CPU-cycle frequency ranges for edge and cloud servers. C3 and C4 limit each edge server's transmission power. C5 and C6 ensure that edge and cloud servers process at most the amount of workloads in their queues. Constraints C7 and C8 guarantee queue stability.

V. Dynamic Workload Power Allocation (DWPA)

We use the concept of bounded optimization to find a solution to P. That is, instead of seeking the minimum as demanded by P directly, we turn to minimizing an upper bound of the minimum. This approach is founded on the principle that by optimizing the upper limit, we can indirectly but effectively optimize P itself.

We divide P into several interconnected subproblems, each corresponding to a part of $\Gamma(t)$, to make it more manageable and allow for tailored solutions. Algorithm 1 overviews the whole process. In the following, we elaborate on the details of each subproblem.

Algorithm 1 DWPA executed at time step t

```
Input: $\{Q_i(t-1)\}_i$, $\{Q_i^{\text{cloud}}(t-1)\}_i$, $\{A_i(t-1)\}_i$, $\{x_{i,j}(t-1)\}_{i,j}$, $\{x_i^{\text{cloud}}(t-1)\}_i$
Output: $\{f_i(t)\}_i$, $\{x_i^{\text{cloud}}(t)\}_i$, $\{p_i^{\text{cloud}}(t)\}_i$, $\{x_{i,j}(t)\}_{i,j}$, $\{p_{i,j}(t)\}_{i,j}$, $\{f_i^{\text{cloud}}(t)\}_i$, $\{Q_i(t)\}_i$, $\{Q_i^{\text{cloud}}(t)\}_i$
$Q_i(t) \leftarrow Q_i(t-1) + A_i(t-1) + \sum_{j \neq i} x_{j,i}(t-1)$ for all $M_i$
Find optimal $f_i(t)$ by (15) for each $M_i$    ▷ P1a
$Q_i(t) \leftarrow Q_i(t) - \text{EC}_i(t)$ for all $M_i$
$Q_i^{\text{cloud}}(t) \leftarrow Q_i^{\text{cloud}}(t-1) + x_i^{\text{cloud}}(t-1)$ for all $M_i$
Find optimal $f_i^{\text{cloud}}(t)$ by solving (16) for each $M_i$    ▷ P1b
$Q_i^{\text{cloud}}(t) \leftarrow Q_i^{\text{cloud}}(t) - CC_i(t)$ for all $M_i$
repeat
    Update $x_{i,j}(t)$ by solving (17) for all $i$ and $j \neq i$    ▷ P2a
    Update $p_{i,j}(t)$ by solving (18) for all $i$ and $j \neq i$    ▷ P2b
until $x_{i,j}(t)$ and $p_{i,j}(t)$ converge
$Q_i(t) \leftarrow Q_i(t) - \sum_{j \neq i} x_{i,j}(t)$ for all $M_i$
repeat
    Update $x_i^{\text{cloud}}(t)$ by solving (20) for all $i$ and $j \neq i$    ▷ P3a
    Update $p_i^{\text{cloud}}(t)$ by solving (21) for all $i$ and $j \neq i$    ▷ P3b
until $x_i^{\text{cloud}}(t)$ and $p_i^{\text{cloud}}(t)$ converge
$Q_i(t) \leftarrow Q_i(t) - x_i^{\text{cloud}}(t)$ for all $M_i$
```

A. Computation Resource Allocation

The first subproblem P1a is to set up each M_i's CPU-cycle frequency, $f_i(t)$, for edge computing to minimize carbon emission $VU_i(t)$ since $U_i(t) = k_i(t)\xi_i f_i(t)^3 T^{\text{cmp}}$. It also affects $\Delta(\Phi(t))$ since it controls $\text{EC}_i(t)$, as indicated by (2), which appears in (9). Therefore, the subproblem is

$$\begin{aligned} \text{P1a} : \min_{f_i(t)} \sum_{i=1}^{s} & \left\{ V k_i(t) \xi_i f_i(t)^3 T^{\text{cmp}} - \frac{Q_i(t) T^{\text{cmp}} f_i(t)}{\phi_i} \right\} \\ \text{s.t. } & 0 \leq f_i(t) \leq f_i^{\max}. \end{aligned} \tag{14}$$

P1a is a convex optimization problem with the optimal solution

$$f_i^*(t) = \sqrt{Q_i(t)/(3Vk_i(t)\xi_i\phi_i)}. \tag{15}$$

If (15) is greater than $f_i^{\max}$, $f_i^*(t)$ is set to $f_i^{\max}$.

Subproblem P1b is to find an optimal $f_i^{\text{cloud}}(t)$ for each i, which affects $VU_i^{\text{cloud}}(t)$ and $Q_i^{\text{cloud}}(t)\text{CC}_i(t)$ in (10).

$$\begin{aligned} \text{P1b} : \min_{f_i^{\text{cloud}}(t)} \sum_{i=1}^{s} & \left\{ V k^{\text{cloud}}(t) \xi^{\text{cloud}} f_i^{\text{cloud}}(t)^3 T^{\text{cmp}} \right. \\ & \left. - \frac{Q_i^{\text{cloud}}(t) T^{\text{cmp}} f_i^{\text{cloud}}(t)}{\phi^{\text{cloud}}} \right\} \\ \text{s.t. } & 0 \leq f_i^{\text{cloud}}(t) \leq f_i^{\text{cmax}}. \end{aligned} \tag{16}$$

Similar to P1a, P1b is convex and the solution is $\sqrt{Q_i^{\text{cloud}}(t)/(3Vk^{\text{cloud}}(t)\xi^{\text{cloud}}\phi^{\text{cloud}})}$, if that value does not exceed f_i^{cmax}, and f_i^{cmax}, otherwise.

B. Workload and Power Selection for Peer Offloading

After setting up $f_i^*(t)$, the resulting $\text{EC}_i(t)$ will be deducted from $Q_i(t)$ for all M_i. The next step is to find out the optimal values, including workload $x_{i,j}(t)$ and transmission power $p_{i,j}(t)$ for all $j \neq i$, for M_i's peer offloading. Both values affect M_i's carbon emission in peer offloading as $U_{i,j}^{\text{off}}(t) = k_i(t)p_{i,j}(t)x_{i,j}(t)/R_{i,j}(t)$ and $R_{i,j}(t)$ is also a function of $p_{i,j}(t)$. Subproblem P2a is to find an optimal $x_{i,j}(t)$ that minimizes $VU_{i,j}^{\text{off}}(t)$ and balances queue lengths assuming a fixed $p_{i,j}(t)$:

$$\begin{aligned} \text{P2a} : \min_{x_{i,j}(t)} \sum_{i=1}^{s} \sum_{j \neq i}^{s} & \{ V k_i(t) p_{i,j}(t) \frac{x_{i,j}(t)}{R_{i,j}(t)} \\ & + (Q_j(t) - Q_i(t)) x_{i,j}(t) \} \\ \text{s.t. } & 0 \leq x_{i,j}(t) \leq x_{i,j}^{\max} \end{aligned} \tag{17}$$

P2a avoids a peer offloading from M_i to M_j if $Q_j(t) > Q_i(t)$. This is because $VU_{i,j}^{\text{off}}(t)$ is non-negative, so $x_{i,j}^*(t) = 0$ yields the minimum value when $Q_j(t) > Q_i(t)$. When $Q_j(t) \leq Q_i(t)$, the optimal $x_{i,j}^*(t)$ depends on the relationship between $a = \frac{Vk_i(t)p_{i,j}(t)}{R_{i,j}(t)}$ and $b = Q_i(t) - Q_j(t)$. It is 0, if $a \geq b$, and $x_{i,j}^{\max} = R_{i,j}(t)T_{i,j}^{\text{off}}$, otherwise. Note that this implies *one-way* peer offloading: $x_{i,j}^*(t) \geq 0 \rightarrow x_{j,i}^*(t) = 0$ for all i, j, $j \neq i$.

Subproblem P2b is to find an optimal transmission power $p_{i,j}^*(t)$ for each $j \neq i$ assuming a fixed $x_{i,j}(t)$ for the same objective.

$$\begin{aligned} \text{P2b} : \min_{p_{i,j}(t)} \sum_{i=1}^{s} \sum_{j \neq i}^{s} & \{ V k_i(t) p_{i,j}(t) T_{i,j}^{\text{off}} \\ & + (Q_j(t) - Q_i(t)) W_i \log_2(1 + \frac{g_{i,j}(t) \times p_{i,j}(t)}{N_0}) T_{i,j}^{\text{off}} \} \\ \text{s.t. } & 0 \leq p_{i,j}(t) \leq p_i^{\max} \end{aligned} \tag{18}$$

With the constraint $0 \leq p_{i,j}^*(t) \leq p_i^{\max}$, it is not difficult to derive the optimal $p_{i,j}^*(t)$ as

$$p_{i,j}^*(t) = \begin{cases} 0, & \text{if } -\frac{b}{a \ln 2} - c \leq 0, \\ -\frac{b}{a \ln 2} - c, & \text{if } 0 < -\frac{b}{a \ln 2} - c < p_i^{\max}, \\ p_i^{\max}, & \text{if } -\frac{b}{a \ln 2} - c \geq p_i^{\max}, \end{cases} \tag{19}$$

where $a = Vk_i(t)T_{i,j}^{\text{off}}$, $b = (Q_j(t) - Q_i(t))W_i T_{i,j}^{\text{off}}$, and $c = N_0/g_{i,j}(t)$.

Unfortunately, $x_{i,j}^*(t)$ and $p_{i,j}^*(t)$ may not be attainable at the same time. We take an iterative approach that works as follows. We first find $x_{i,j}^*(t)$ assuming an arbitrary $p_{i,j}(t)$ and then find $p_{i,j}^*(t)$ assuming an arbitrary $x_{i,j}(t)$. We are done if both $|x_{i,j}^*(t) - x_{i,j}(t)|$ and $|p_{i,j}^*(t) - p_{i,j}(t)|$ are relatively small. Otherwise, we repeat the process to find new optimal values assuming $p_{i,j}^*(t)$ and $x_{i,j}^*(t)$. This process iterates a limited number of times.

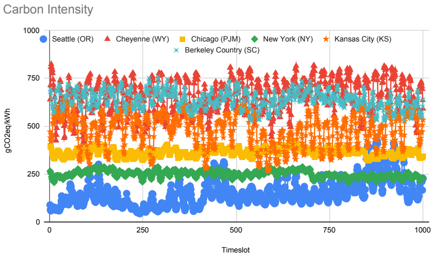


Figure 3: Carbon intensities in Berkeley County, Seattle, Cheyenne, Chicago, New York, and Kansas City from 00:00, July 1, 2023, to 15:00, August 11, 2023 [7]. Each time slot was one hour.

C. Workload and Power Selection for Vertical Offloading

After deducting $\sum x_{i,j}(t)$ from $Q_i(t)$, the next step is to find M_i's optimal setting for vertical offloading. Similar to peer offloading, the goal is to minimizes $VU^{\text{off}}_{i,c}(t)$ yet balances the queue lengths between $Q_i(t)$ and $Q^{\text{cloud}}_i(t)$. The setting includes an optimal $x^{\text{cloud}}_i(t)$ value assuming a fixed $p^{\text{cloud}}_i(t)$

$$\begin{aligned} \text{P3a}: \min_{x^{\text{cloud}}_i(t)} & \sum_{i=1}^{s} \{Vk_i(t)p^{\text{cloud}}_i(t)\frac{x^{\text{cloud}}_i(t)}{R_{i,c}(t)} \\ & + (Q^{\text{cloud}}_i(t) - Q_i(t))x^{\text{cloud}}_i(t)\} \\ & \text{s.t. } 0 \le x^{\text{cloud}}_i(t) \le x^{\text{cmax}}_i \end{aligned} \tag{20}$$

and an optimal $p^{\text{cloud}}_i(t)$ value assuming a fixed $x^{\text{cloud}}_i(t)$

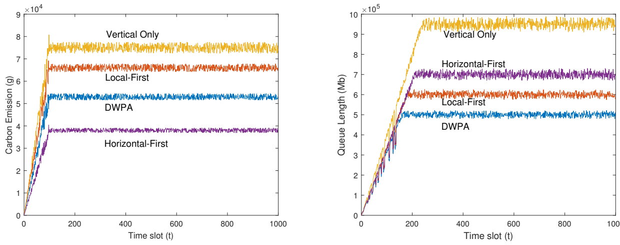
$$\begin{aligned} \text{P3b}: \min_{p^{\text{cloud}}_i(t)} & \sum_{i=1}^{s} \{Vk_i(t)p^{\text{cloud}}_i(t)T^{\text{off}}_{i,c} \\ & + (Q^{\text{cloud}}_i(t) - Q_i(t))W_i \log_2(1 + \frac{g_{i,c}(t) \times p^{\text{cloud}}_i(t)}{N})T^{\text{off}}_{i,c}\} \\ & \text{s.t. } 0 \le p^{\text{cloud}}_i(t) \le p^{\text{cmax}}_i. \end{aligned} \tag{21}$$

Problem P3a is similar to P2a. The objective function is convex and the optimal value of $x^{\text{cloud}}_i(t)$ is $R_{i,c}(t)T^{\text{off}}_{i,c}$, if $Vk_i(t)p^{\text{cloud}}_i(t)/R_{i,c}(t) < Q_i(t) - Q^{\text{cloud}}_i(t)$, and zero, otherwise. Similarly, the solution to P3b is the same as shown in (19) with a, b, and c replaced by $Vk_i(t)T^{\text{off}}_{i,c}$, $(Q^{\text{cloud}}_i(t) - Q_i(t))W_iT^{\text{off}}_{i,c}$, and $N_0/g_{i,c}(t)$, respectively. An iterative approach is also used to find a converged solution.

VI. Performance Evaluation

We assumed a cloud center in Berkeley County, USA, and five edge servers in Seattle, Cheyenne, Chicago, New York, and Kansas City, respectively. We used 1000-hour historical data [7] for CIs in these cities (Fig. 3).

We simulated 1000 hours (time slots). We used an exponential moving average to estimate the CI during each hour at the beginning of the time slot. We used the On-Off model to generate workload traffic. The mean values in the On and Off states were 64 MB and 3.2 MB, respectively, with standard deviation 2 MB. The state transition probabilities from On to Off and from Off to On states were 0.25 and 0.15, respectively. Other parameters were $f^{\max}_i = 10$ GHz, $f^{\text{cmax}}_i = 80$ GHz, $\phi_i = \phi^{\text{cloud}}_i = 1000$, $\xi_i = \xi^{\text{cloud}}_i = 10^{-18}$, $p^{\max}_i = 1$ W, $N_0 = -130$ dBm, and $W_i = 1$ MHz.



(a) Carbon emission

(b) Queue length

Figure 4: Comparisons of DWPA with its variants

We investigated the performance of DWPA in terms of carbon emissions and queue length. The results were compared with other approaches, including three variants of DWPA (Local-First, Vertical-Only, and Horizontal-First) and three counterparts (ICSOC-19, DOLA-22, and YCL-24).

A. DWPA and Its Variants

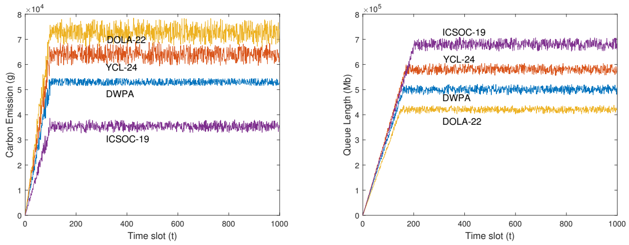
Local-First processes workload locally and offloads the workload to the cloud only when the local server queue is full. Vertical-Only turns off the option of peer offloading. Horizontal-First permits vertical offloading only when peer offloading is impossible (i.e., all outbound peer servers' queues are full). As shown in Fig. 4a, Vertical-Only generated the highest carbon emissions and the longest queue lengths. This result serves as a baseline and helps in understanding the benefits of peer offloading. Local-First outperformed Vertical-Only because it hardly offloaded workload to the cloud, and most edge servers had lower CI values than the cloud server. DWPA exhibited lower carbon emissions than Local-First and Vertical-Only, thanks to the option of peer offloading. The lowest carbon emissions by Horizontal-First also confirmed the benefit of peer offloading in carbon reduction.

While peer offloading reduced carbon footprint, local computation and vertical offloading could help balance queue backlogs. This can be seen from Fig. 4b, where Horizontal-First had more considerable queue lengths than Local-First and DWPA. DWPA had the smallest queue lengths due to the flexibility of the mixed-use of local computing and horizontal/vertical offloading.

B. DWPA and Its Counterparts

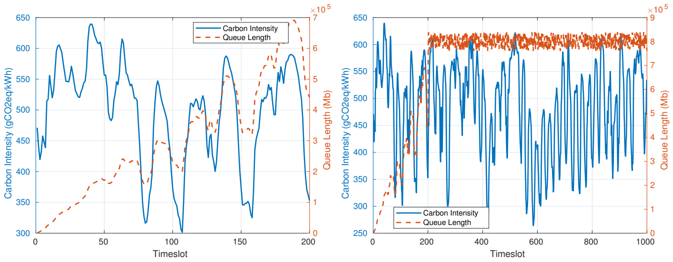
We considered three counterparts of DWPA. ICSOC-19 [16] offloads workload to the server with the lowest CI for processing, disregarding the servers' queue lengths. DOLA-22 [17] allocates workload in proportion to queue backlogs. YCL-24 [15] assumes static CI and considers vertical offloading only.

As shown in Fig. 5, ICSOC-19 produced the lowest carbon footprint but the largest queue lengths due to its greedy design. On the other hand, DOLA-22 had the smallest queue lengths but also the highest carbon footprint. Although both YCL-24 and DWPA balanced these two performance metrics, DWPA outperformed YCL-24 due to its ability to exploit horizontal offloading.



(a) Carbon emission (b) Queue length

Figure 5: Comparisons of DWPA with the counterparts



(a) The first 200 hours (b) The whole 1000 hours

Figure 6: The relationship between carbon intensity and queue length in Kansas City

C. Balancing Carbon Emissions And Queue Lengths

We illustrate how DWPA balances carbon emissions and queue backlogs by closely examining the activities in Kansas City. Fig. 6 show the carbon emission and queue length in the first 200 and the whole 1000 hours. In the first 200 hours, the queue length did not stabilize and went with the CI value. However, the queue length was not affected by CI once it stabilized.

VII. Conclusions

We have proposed DWPA, an online workload and power configuration scheme for vertical/horizontal offloading in a collaborative edge-cloud system. DWPA balances carbon footprint and queue stability by allocating edge and cloud computing resource and configuring workload and transmission power for peer and vertical offloading. Simulation results based on historical CI data show that, compared with its counterparts, DWPA can improve carbon footprint and queue lengths due to its flexibility in offloading directions.

Acknowledgment

This work was supported by the National Science and Technology Council of Taiwan under grant numbers NSTC 113-2221-E-A49-180 and NSTC 114-2218-E-A49-018.

References

[1] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: Multiaccess edge computing for 5G and Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6722–6747, 2020.

[2] E. Ahvar, A.-C. Orgerie, and A. Lebre, "Estimating energy consumption of cloud, fog, and edge computing infrastructures," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 2, pp. 277–288, 2022.

[3] C. Jiang, T. Fan, H. Gao, W. Shi, L. Liu, C. Cérin, and J. Wan, "Energy aware edge computing: A survey," *Comput. Commun.*, vol. 151, pp. 556–580, 2020.

[4] L. Belkhir and A. Elmeligi, "Assessing ICT global emissions footprint: Trends to 2040 & recommendations," *J. Clean. Prod.*, vol. 177, pp. 448–463, 2018.

[5] Y. Xiao, G. Shi, Y. Li, W. Saad, and H. V. Poor, "Toward self-learning edge intelligence in 6G," *IEEE Commun. Mag.*, vol. 58, no. 12, pp. 34–40, 2020.

[6] C. Qu, M. Tao, and R. Yuan, "A hypergraph-based blockchain model and application in internet of things-enabled smart homes," *Sensors*, vol. 18, no. 9, 2018.

[7] "Carbon intensity map," https://app.electricitymaps.com, accessed: 2025-03-27.

[8] Y. Mao et al., "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, 2017.

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

[10] M. P. J. Mahenge, C. Li, and C. A. Sanga, "Energy-efficient task offloading strategy in mobile edge computing for resource-intensive mobile applications," *Digit. Commun. Netw.*, vol. 8, no. 6, pp. 1048–1058, 2022.

[11] Y. Chen, J. Xu, Y. Wu, J. Gao, and L. Zhao, "Dynamic task offloading and resource allocation for NOMA-aided mobile edge computing: An energy efficient design," *IEEE Trans. Serv. Comput.*, vol. 17, no. 4, pp. 1492–1503, 2024.

[12] Y. Li, S. Xia, M. Zheng, B. Cao, and Q. Liu, "Lyapunov optimization-based trade-off policy for mobile cloud offloading in heterogeneous wireless networks," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 491–505, 2022.

[13] S. Sorrell, "Jevons' paradox revisited: The evidence for backfire from improved energy efficiency," *Energy Policy*, vol. 37, no. 4, pp. 1456–1469, 2009.

[14] K. Chen, Y. Sun, S. Zheng, H. Yang, and P. Yu, "Online collaborative energy-network resource scheduling for WPT-enabled green edge computing," *IEEE Trans. Green Commun. Netw.*, vol. 8, no. 2, pp. 601–618, 2024.

[15] Y. Yang, Y. Chen, K. Li, and J. Huang, "Carbon-aware dynamic task offloading in NOMA-enabled mobile edge computing for IoT," *IEEE Internet Things J.*, vol. 11, no. 9, pp. 15 723–15 734, May 2024.

[16] P. Lai, Q. He, G. Cui, X. Xia, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Edge user allocation with dynamic quality of service," in *Proc. Int. Conf. Service-Oriented Comput.*, 2019, pp. 86–101.

[17] J. Huang, M. Wang, Y. Wu, Y. Chen, and X. Shen, "Distributed offloading in overlapping areas of mobile-edge computing for internet of things," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13 837–13 847, 2022.