Task-Aware Offloading for Heterogeneous DC-HAP Systems

Jichen Lu, Balsam Alkouz, Osama Amin, Basem Shihada
King Abdullah University of Science and Technology (KAUST)
Thuwal, Makkah Prov., 23955, Saudi Arabia
{jichen.lu, balsam.alkouz, osama.amin, basem.shihada}@kaust.edu.sa

***Abstract*—Data center-enabled High Altitude Platforms (DC-HAPs) act as a promising green computing service provider for data centers by utilizing stratospheric cooling and solar power. This paper investigates advanced service routing through task-aware offloading strategies aimed at reducing the energy consumption of ground data centers. We employ queueing models for both transmission and computation processes. Then, we develop a heuristic algorithm with the derived formula to solve the task-wise offloading problem, considering task type awareness, topology heterogeneity, and server-level heterogeneity. Simulation studies demonstrate that our method consistently surpasses traditional approaches in both efficiency and energy saving performance under different complex system configurations. These results support the practical adoption of DC-HAPs for next-generation sustainable Compute-as-a-Service (CaaS) providers.**

***Index Terms*—DC-HAP, sustainable CaaS, non-terrestrial network, task-aware offloading, queue delay.**

I. Introduction

The rapid expansion of digital services, including cloud computing and large-scale data analytics, has led to a dramatic increase in data center energy consumption [1], [2]. With the proliferation of connected devices and applications, forecasts indicate that data centers could consume as much as 10% of the world's electricity by 2050 [3]. Therefore, integrating energy efficiency as a core Service-Level Objective (SLO) has become essential [4]. Sustainability is increasingly emerging as a key Quality-of-Service (QoS) requirement in recent services, alongside traditional performance constraints such as latency and service drop ratio [5]–[8].

The inclusion of sustainability as a QoS dimension motivates a complementary role between aerial and terrestrial infrastructures. Conventional Terrestrial Data Centers (TDCs) act as *service consumers* with substantial energy overhead, primarily due to cooling and computational processing. In contrast, aerial-based Data Center-enabled High Altitude Platforms (DC-HAPs) can operate as *service providers*, offering Compute-as-a-Service (CaaS) with *inherent sustainability advantages* [9]–[11]. Positioned in the stratosphere, DC-HAPs benefit from naturally low temperatures and abundant solar energy, significantly reducing cooling demands and enabling the use of renewable power. TDCs can leverage DC-HAPs to *route* computation-intensive and energy-demanding services. This service federation between TDCs and DC-HAPs enables dynamic workload sharing that balances performance, capacity, and sustainability [12].

Service routing between TDCs and DC-HAPs naturally manifests as a task offloading problem. The current TDCs to DC-HAPS task offloading solutions remain limited for three main reasons: (1) they typically deal with a *single task type* and thus fail to capture workload diversity [11]; (2) they focus on *one-to-one offloading* between a single TDC and a single HAP [13], whereas practical deployments will involve many-to-many interactions; and (3) they assume *homogeneous infrastructures*, where TDCs and HAPs have identical topologies and the same number of servers per node [11].

To overcome these limitations, we present a task offloading framework for federated DC-HAP systems that jointly optimizes communication and computation constraints while explicitly addressing *workload diversity*, *infrastructure heterogeneity*, and *server-level heterogeneity*. Figure 1 illustrates the architecture of our proposed task-aware offloading framework for DC-HAP systems. We summarize our main contributions as follows:

- We propose a task-aware offloading framework for DC-HAP systems that jointly models transmission and computation while supporting multiple task types.
- We design a heuristic algorithm with a novel indicator that achieves robust convergence and adaptability, and compare it against Reinforcement Learning (RL) and throughput-based baselines.
- We evaluate the framework under realistic scenarios, including dynamic loads, topology heterogeneity, and server-level heterogeneity, demonstrating superior utilization and stability.

II. Related Work

DC-HAPs have emerged as a promising infrastructure for next-generation service delivery in 6G and beyond, offering computing and communication capabilities that complement terrestrial systems [14]. By operating in the stratosphere, DC-HAPs benefit from wide-area coverage, stable line-of-sight links, and favorable environmental conditions that reduce cooling energy consumption and enable renewable energy utilization [9]. Prior studies have demonstrated that DC-HAPs can support ultra-low-latency and computation-intensive services in dense urban environments, such as intelligent transportation systems [15], while also providing resilient computing and connectivity in remote or disaster-stricken regions. These properties position DC-HAPs as sustainable service providers that can enhance scalability, reliability, and energy efficiency when integrated with terrestrial data centers.

From a service computing perspective, most service federation and task offloading studies focus on cloud-to-edge

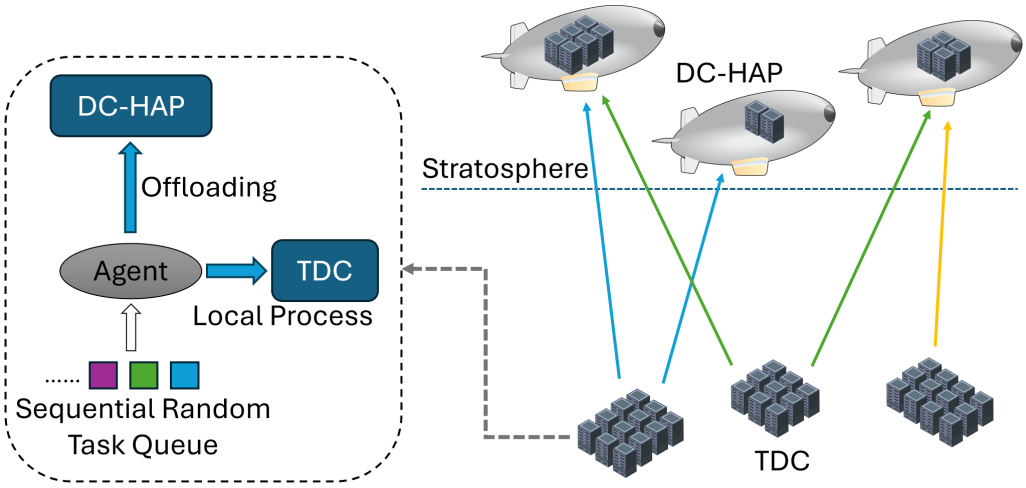


Fig. 1: Proposed service-aware routing framework for DC-HAP systems.

(cloud-to-MEC) paradigms, mainly driven by latency reduction and user proximity [16]–[18]. These works prioritize service placement, edge caching, elastic offloading, and mobility-aware resource management to meet strict delay, reliability, and quality-of-service requirements in vehicular, IoT, and immersive applications [19]–[23]. Other Compute-as-a-Service (CaaS) federation models include cloud-to-cloud federation for load balancing and fault tolerance [24], [25], multi-cloud interoperability to mitigate vendor lock-in [26], [27], and geo-distributed clouds to improve availability and resilience [28]–[30]. In contrast, our work shifts the focus from latency-driven federation to sustainability-driven federation by TDCs with DC-HAPs.

III. System Modeling

To enable rigorous analysis of task offloading in DC-HAP systems, it is essential to establish clear models for both computation and communication. In this work, we concentrate on uplink transmission and HAP computation processes, under the assumption that the computational results generated on DC-HAPs are relatively small and can be efficiently transmitted back to the source [31], [32]. Building on this foundation, the remainder of this section introduces the computation model, the wireless communication model, and the offloading delay model.

A. Computation Model

We begin by introducing the computation model, which defines the computational components, their processing capabilities, and task characteristics.

Consider a system with TDCs and DC-HAPs; both can undertake computational tasks. Assume that for each HAP, it has $N_{\mathrm{s},m}^{\mathrm{HAP}}$ servers, each with a computing capacity of μ Instructions Per Second (IPS). The tasks arrive at TDCs first and can be processed locally or at DC-HAPs after offloading. For each task $i \in 1, \ldots, N$, we define three key parameters: instruction length l_i (instructions/task) representing the number of instructions needed for processing the task; bit length b_i (bits/task) representing the data size required for transmission; and delay constraint τ_i.

Following [11], [13], the energy saved in the TDCs is inversely proportional to the total utilization of the DC-HAPs. Since now we involve the consideration of heterogeneous HAP N_{s}, we propose to use a normalized summation of the DC-HAPs' utilization as our optimization objective:

$$\rho^{\mathrm{sum}} = \sum\nolimits_{m=1}^{N_{\mathrm{HAP}}} \frac{N_{\mathrm{s},m}^{\mathrm{HAP}}}{N_{\mathrm{s,max}}^{\mathrm{HAP}}} \frac{\bar{t}_{\mathrm{busy},m}}{t_{\mathrm{total},m}}, \tag{1}$$

where the N_{HAP} is the total number of HAPs in the system; $N_{\mathrm{s},m}^{\mathrm{HAP}}$ is the number of servers in the HAP m; $N_{\mathrm{s,max}}^{\mathrm{HAP}}$ is the max number of servers in the HAP limited by the payload; $t_{\mathrm{total},m}$ is the evaluation period, and $\bar{t}_{\mathrm{busy},m}$ is the average busy period of all servers in the HAP.

B. Wireless Communication Model

We assume a MIMO channel is used for the communication between the TDC and the HAP. From [13], we apply the achievable capacity for the channel as:

$$R_{\min} = B \log\left(1 + N_{\mathrm{r}}\gamma\right). \tag{2}$$

where B is the channel bandwidth. γ is the Free Space Path Loss (FSPL) defined as [13], [33].

C. Offloading Delay Model

We assume a queue model for both the transmission and computational process of the offloading [13], while applying the First-in-First-out (FIFO) queueing strategy [34]. For the transmission stage and the computational stage, the time can be divided into the service time and the waiting time in the queue, which gives the total sojourn time [13]:

$$T = W_{\mathrm{trans}} + S_{\mathrm{trans}} + W_{\mathrm{comp}} + S_{\mathrm{comp}} \tag{3}$$

For task i, if the sojourn time is larger than its delay constraint, as $T_i > \tau_i$, it will be dropped.

IV. Task-aware Offloading Problem Formulation

We analyze the DC-HAP system's resource utilization based on the delay model developed in the previous section. Here, we first consider a simple scenario: a single TDC-HAP and a task queue with a single type. Then, after the analysis of it, we will extend to a more general problem, which is task-wise decision making.

A. Queue-wise Optimization Problem

When considering a queue with a certain type of task offloading in a single TDC-HAP system, we optimize the offloading task rate λ of the type to gain maximum single HAP utilization. Previous work has established the maximum offloaded task rate considering a latency constraint in [13, Theorem 1]. The λ to gain maximum ρ:

$$\tilde{\lambda}_{\max} = \frac{-B - \sqrt{B^2 - 4AC}}{2A} \tag{4}$$

where A, B, C are:

$$\begin{aligned} A = {} & 2\left(\tau - \frac{b}{R} - \frac{l}{\mu}\right)\frac{b}{R}\frac{l}{N_{\mathrm{s}}^{\mathrm{HAP}}\mu} + \left(1 + c_{\mathrm{s,trans}}^2\right) \\ & \frac{b^2}{R^2}\frac{l}{N_{\mathrm{s}}^{\mathrm{HAP}}\mu} + \left(1 + c_{\mathrm{s,comp}}^2\right)\frac{l^2}{(N_{\mathrm{s}}^{\mathrm{HAP}})^2\mu^2}\frac{b}{R} \end{aligned} \tag{5a}$$

$$\begin{aligned} B = {} & 2\left(\tau - \frac{b}{R} - \frac{l}{\mu}\right)\left(-\frac{b}{R} - \frac{l}{N_{\mathrm{s}}^{\mathrm{HAP}}\mu}\right) \\ & - \left(1 + c_{\mathrm{s,trans}}^2\right)\frac{b^2}{R^2} - \left(1 + c_{\mathrm{s,comp}}^2\right)\frac{l^2}{(N_{\mathrm{s}}^{\mathrm{HAP}})^2\mu^2} \end{aligned} \tag{5b}$$

$$C = 2\left(\tau - \frac{b}{R} - \frac{l}{\mu}\right). \tag{5c}$$

where $c_{\mathrm{s,trans}}, c_{\mathrm{s,comp}}$ are the coefficient of variation of the service time for the two stages. Then, we get the maximum achievable computational utilization as:

$$\rho = \frac{\lambda_{\max} l}{N_{\mathrm{s}}^{\mathrm{HAP}}\mu} \tag{6}$$

B. General Task-wise Problem Formulation

Although 6 gives us a closed-form expression for the achievable performance of the system, it can not be well generalized since different types of tasks are mixed in general situations. Therefore, for general practical deployment, we need to decide the offloading action for each incoming task i at time t_i, with bit length b_i and instruction length l_i.

We aim to maximize computational utilization ρ^{sum} for HAPs, while maintaining delay constraint for each task. Our optimization variables are the action a_i for each task i. Therefore, we can formulate the task-wise problem as follows:

$$\max_{a_i} \rho^{\mathrm{sum}} \tag{7a}$$

$$\text{s.t. } T_i(a_i) \le \tau_i, \forall i \in [1, N] \tag{7b}$$

$$a_i \in \{0, 1, 2, \ldots, N_{\mathrm{HAP}}\}, \forall i \in [1, N] \tag{7c}$$

where N is the total number of tasks, N_{HAP} is the number of HAPs, and τ_i is the maximum delay for task i. The sojourn time $T_i(a_i)$ is the time taken to process task i with action a_i, which is a function of the offloading action a_i.

Note that the objective (7a) has a constant denominator when the number of HAPs and servers are fixed, so we can ignore it in the optimization. The problem has delay constraints (7b) for each task, and a discrete action space, which makes it hard to solve with traditional optimization methods.

V. Multi-Type Task Offloading in Heterogeneous DC–HAP Systems

Previous works have proposed algorithms optimizing the offloading task rate for different task queues [13], [35]. In this work, we construct algorithms for task-wise offloading under general heterogeneous systems.

The majority of methods to handle task-wise decision making are RL algorithms [36]–[38]. Although the RL algorithm is powerful, it requires an extra computational resource and time for model training. At the same time, it is difficult to converge when the number of nodes in the system is large. Hence, we propose a heuristic algorithm that is based on the previously derived indicator ρ in 6, which maintains both the flexibility to arbitrary task properties and the efficiency.

A. Task-wise Reward Function

Since we need to consider task-wise decision-making, a fine-grained feedback is needed to each decision made. Here, we borrow the reward function idea from RL algorithms [36], [39], and make modifications for our problem.

Considering the delay constraints, we need to add a penalty term to the reward function when the task exceeds the delay constraint. At the same time, for the tasks that were successfully processed in the HAPs, a reward should be applied to reflect the energy it saved for the TDC. As for the tasks finished in the TDCs, they didn't contribute to our energy saving target, thus should be assigned a minor reward if they are successfully finished. Therefore, we define the reward function as follows:

$$r = \begin{cases} \begin{cases} l_0, & \text{if success} \\ -l_0, & \text{if dropped} \end{cases}, & \text{if } a_{m,i} = 0, \\ \begin{cases} l_i, & \text{if success} \\ -l_i * p, & \text{if dropped} \end{cases}, & \text{if } a_{m,i} \neq 0. \end{cases} \tag{8}$$

B. Adjusted Preference Indicator

We utilize the derived analytical results ρ in 6 to indicate the probability of offloading: if offloading can get higher expected ρ, then it has a higher probability.

Considering multiple TDC-HAP scenarios, we propose to deploy distributed algorithms for each TDC to enable flexibility of the algorithm. Assume we have the throughput matrix $\mathbf{R}^{N_{\mathrm{TDC}} \times N_{\mathrm{HAP}}}$, if task i is generated in TDC n, then we use $R[n, :]$ to calculate vector ρ. To increase the adaptation performance of the algorithm for different topologies, we propose to use the power factor p to adjust the preference indicator ρ. The adjusted indicator vector is then expressed as ρ^p, where the power factor p can influence the sensitivity of the algorithm to the ρ, as in line 10.

In addition to our indicator ρ, we also prepare two indicators as baselines: 1) an identical $\mathbf{1}/N_{\mathrm{HAP}}$, which is represented as *random* since it has equal preference; 2) normalized throughput-based preference $\mathbf{R}/||\mathbf{R}||$, which makes the decisions based on the throughput to the HAPs.

Algorithm 1 Heuristic Algorithm for Task-wise Offloading

1: **Input:** $\mathbf{b}, \mathbf{l}, \tau$: task property vectors with size N; $\mathbf{R}$: transmission throughput matrix; μ: computation capacity for each server; $N_{\mathrm{s},m}^{\mathrm{HAP}}$: number of servers on the HAP m; N_{HAP}: number of HAPs.
2: **Output:** $a_i \in \{0, 1, 2, \ldots, N\}$ for each task.
3: **Initialization:** Initialize: $\alpha = 1.0$; $p = 1.0$; $\Delta_\alpha = 0.2$; $\Delta_p = 0.2$. Initialize the action-reward history queue $Q = \{\}$ with fixed size K. Initialize the periodic reward-parameter history queue $Q_\theta = \{\}$.
4: **Begin:**
5: **for** task i **do**
6: Calculate the ρ_{i} using (6). Sample $s_2 \sim U(0, 1)$
7: **if** $s_1 < \alpha$ **then**
8: Process the task locally, i.e., $a_i = 0$.
9: **else**
10: Get the adjusted probability $P_i = (\rho_i)^p$; sample index $j \sim U(1, N_{\mathrm{HAP}})$, $s_2 \sim U(0, 1)$.
11: **if** $s_2 < P_i[j]$ **then**
12: Offload the task to HAP j, i.e., $a_i = j$.
13: **else**
14: Process the task locally, i.e., $a_i = 0$.
15: **end if**
16: **end if**
17: Record the reward r_i using (8) to Q.
18: **if** $|Q| \geq L$ **then**
19: $\bar{r} = \mathrm{mean}(Q)$; Clear $Q = \{\}$.
20: **if** $\bar{r} > \max(Q_\theta(\bar{r}))$ **then**
21: $\alpha_r = \alpha$, $p_r = p$.
22: **end if**
23: Append $(\bar{r}, \alpha, p)$ to Q_θ.
24: $\alpha = \alpha_r + \Delta_\alpha * U(-1, 1)$, $p = p_r + \Delta_p * U(-1, 1)$.
25: Cut α and p as: $\alpha \in [0, 1]$, $p \in [0.1, 5.0]$.
26: Update step with decay $\Delta_\alpha = \beta\Delta_\alpha$, $\Delta_p = \beta\Delta_p$.
27: **end if**
28: **end for**
29: END.

C. Sampling-based Decision Making

The details of our algorithm is shown in Algorithm 1. The algorithm is sampling-based with the help of our constructed indicator ρ^p, and has a local sample ratio factor α.

For each incoming task i, we first random sample $s_1 \sim U(0, 1)$, and process i locally if $s_1 < \alpha$ as line 8. This step of the statistical filter can reduce the possible congestion when a large task-generating rate is considered. If $s_1 \geq \alpha$, we sample $j \sim U(1, N_{\mathrm{HAP}})$ and $s_2 \sim U(0, 1)$. If $s_2 < \rho^p[j]$, then we offload i to HAP j, otherwise we process it locally, as from line 9 to line 16.

D. Hyperparameter Tuning

To find the optimal values for hyperparameters α and p, we apply a stochastic search. We record the reward of each decision in queue Q. Whenever $|Q| == L$, we calculate the average reward as $\bar{r}$, as line 19. Then, we compare the current $\bar{r}$ with maximum $\bar{r}$ in the parameter history queue Q_θ: if the current average reward is higher, we replace the reserved α_r, p_r with α, p, as line 21. Finally, we do a stochastic move for α, p as line 24, and do step decay in 26.

TABLE I: System Parameters

Parameter	Symbol	Value / Range
HAP altitude	L_{HAP}	20 km
Servers per HAP	$N_{s,\mathrm{HAP}}$	40 *Varied in Exp. 4
Servers per TDC	$N_{s,\mathrm{TDC}}$	200 *Varied in Exp. 4
Workload of TDC	ρ_{TDC}	30% [40]*Varied in Exp. 2
Server capacity	μ	580 MIPS [11]
Tx power (TDC)	P_{trans}	100 W
Tx antennas (TDC)	N_t	2
Carrier frequency	f_c	31 GHz
Bandwidth	B	100 MHz
Rx antennas (HAP)	N_r	16
Task data size	b	300–800 KB [41]
Task compute load	l	100–1000 MCycles [41]
Cycles-to-instructions ratio	–	3.7 [41]
Latency constraint	τ	500–1000 ms

VI. Simulation Results

We evaluate the framework under multiple HAP–TDC settings. The common approach to solve this problem is to use reinforcement learning (RL) methods [36]–[39]. We employ the Deep Q-Network (DQN) as our reinforcement learning baseline, as it is well-suited for discrete action spaces. For fairness, we will use the same reward function for the RL baseline during simulation experiments. In addition, we provide two baselines for our heuristic algorithm architecture: 1) randomly select the HAP to upload; 2) select the HAP based on the throughput from the TDC.

A. Simulation Setup and Experiments

We consider a **multi-HAP to multi-TDC system**. The positions of TDCs and HAPs are randomly generated within a square of 50*50 km. The positions are consistent for each series of experiments to ensure the fairness of comparison. We simulate the experiments on a single Intel Cascade Lake CPU with 2.5GHz and 16GB of memory. The simulation is designed around four experiments as follows:

- **Experiments 1 and 2 (Baseline Convergence and Dynamicity):** We fix the number of HAPs and TDCs to 4 and 4. The setting aims to measure the algorithm *convergence* and robustness under *dynamic variations*.
- **Experiment 3 (Topology Heterogeneity):** We investigate heterogeneity in terms of *system topologies* by varying the number of HAPs and TDCs.
- **Experiment 4 (Server-Level Heterogeneity):** Finally, we extend heterogeneity to the infrastructure layer by varying the number of servers within both HAPs and TDCs. This introduces variability not only in topology but also in computational capacity.

Note that the task data size and computational load are randomly drawn in all experiments, ensuring that the evaluation naturally covers the challenge of handling *different task types*. Moreover, the use of multiple transmitting and receiving antennas enables multi-directional communication, supporting the *many-to-many offloading* relationships central to practical DC-HAP deployments. The main system parameters used across all experiments are summarized in Table I.

B. Results and Discussion

*1) **Convergence Analysis**:* By convergence, we refer to the ability of an algorithm to stabilize its performance after

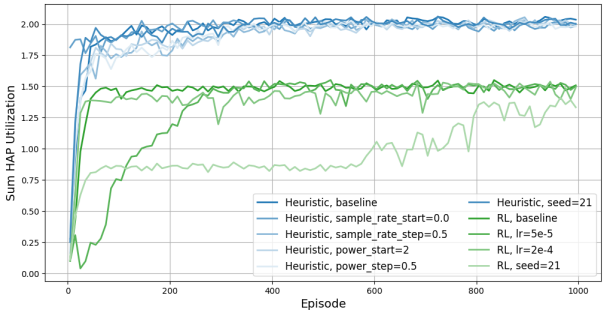


Fig. 2: Convergence analysis

a sufficient number of iterations, regardless of the choice of initial hyperparameters. This property is essential to ensure reliability under diverse deployment conditions.

As shown in Figure 2, the algorithms are tested under different random seeds, learning rates, and initializations. The proposed heuristic consistently converges to a stable utilization level, demonstrating strong robustness. In contrast, while the RL-based baseline also converges, it exhibits slower stabilization, lower final utilization, and higher variance, highlighting its reduced reliability in practice.

*2) **Dynamic Adaptation**:* By dynamic adaptation, we refer to an algorithm's ability to maintain stable performance when system conditions fluctuate over time. Such resilience is critical since TDC utilization rarely remains constant.

Figure 3 illustrates this behavior across three layers: the bottom subplot shows the scheduled increase in TDC utilization at each timestep, the middle subplot reports the resulting task drop ratio, and the top subplot depicts the HAP utilization achieved under different algorithms. The proposed heuristic algorithm maintains both high utilization and a near-zero drop ratio despite the shifting load, while the RL algorithm still remains around a 2% drop ratio. The throughput-based heuristic performs adequately but deteriorates once computational congestion dominates, since the increasing computational service waiting time is not considered. In contrast, the RL-based baseline suffers the most, with frequent utilization drops and higher task loss whenever the TDC utilization changes, underscoring its limited adaptability to dynamic environments. The main reason is that the RL agents' exploration rate is low when they converge to a stable state, which leads to the failure of adapting to new conditions.

*3) **Topology Heterogeneity**:* By topology heterogeneity, we refer to changes in the system structure caused by varying the number of TDCs and HAPs. Such scenarios capture practical deployments where the distribution and arrangement of computing resources cannot be assumed fixed.

We first fix the number of HAPs at four and vary the number of TDCs from 1 to 8. Figure 4 shows the resulting HAP utilization, where the proposed heuristic with the ρ indicator consistently achieves the highest utilization across all topologies. As the number of TDCs increases, the sum utilization increases due to: 1) more workloads are considered; 2) new TDCs may involve higher transmission throughput. The corresponding drop ratios remain zero for up to four TDCs. When the system scales to eight TDCs, ρ slightly increases the drop ratio compared to the R indicator

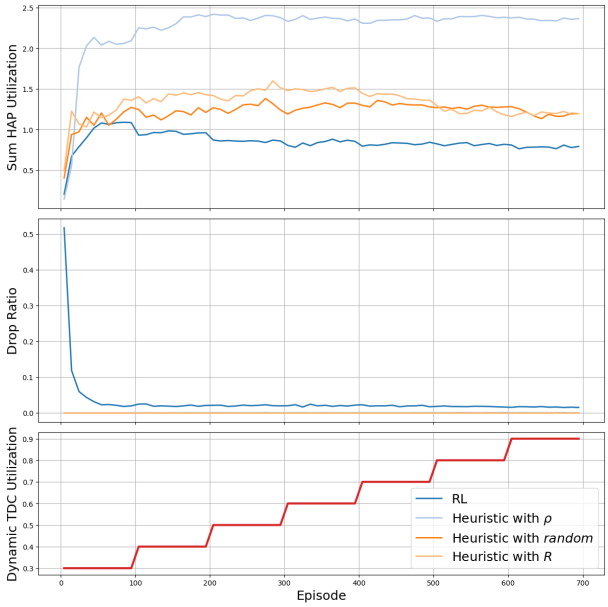


Fig. 3: Dynamic adaptation

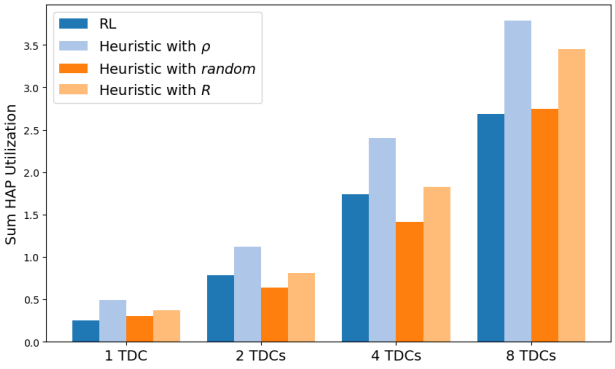


Fig. 4: Sum HAP utilization for varying number of TDCs

but maintains significantly higher utilization as a trade-off, and our heuristic algorithm still gets a lower drop ratio than the RL method. In contrast, the RL baseline delivers both the lowest utilization and the highest drop ratios.

Next, we fix the number of TDCs at four and vary the HAPs. As shown in Figures 5, our ρ-based heuristic outperforms all alternatives. This demonstrates the benefit of jointly considering transmission and computation loads. Note that the sum HAP utilization does not rise after 4 HAPs, which is caused by the low general workload of TDCs and the bottleneck of throughput limitation of transmitters. We also observe that the drop ratio of the heuristic algorithms remains zero except increases to around 1% when the number of HAPs is 1 and 2, but still lower than the RL baseline. Similar to the aforementioned observation, this suggests that congestion will occur in HAP servers when the number of TDCs is much larger than the number of HAPs and multiple incoming offloading links are considered. This highlights the importance of balancing the HAP-to-TDC ratio. Such behavior gives a practical insight into the appropriate deployment density of HAPs in real-world scenarios.

Finally, Figures 6a and 6b report the efficiency results. Figure 6a shows the inference time cost for each decision-making. The ρ-based heuristic requires slightly more time as

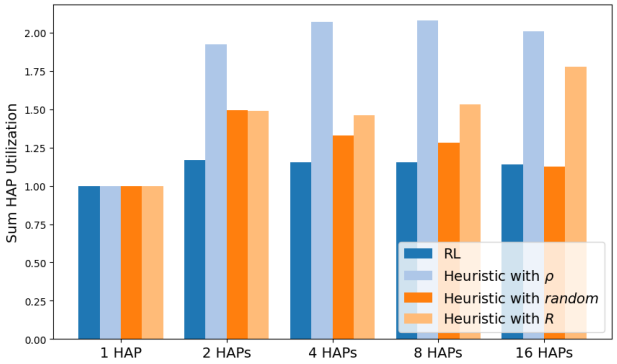
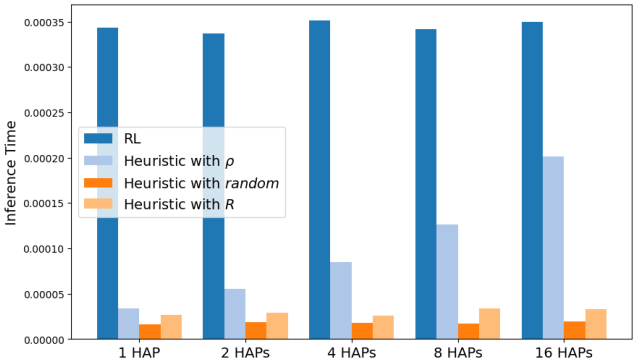
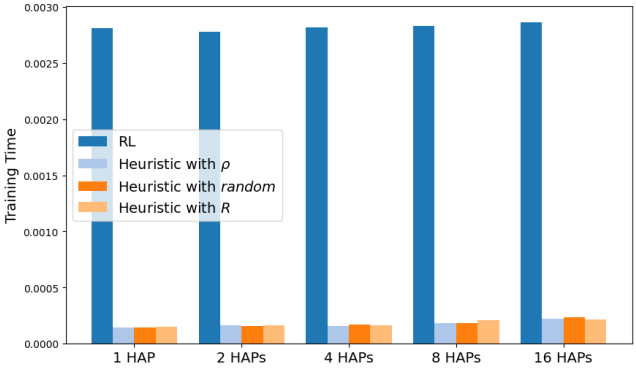


Fig. 5: Sum HAP utilization for varying number of HAPs



(a) Inference time



(b) Training time

Fig. 6: Time cost for varying number of HAPs

the number of HAPs increases, since it explicitly evaluates both transmission and computation states, but its overhead remains very low compared to RL. The throughput and random-based heuristics have constant inference time since they rely only on direct observations. However, as shown in Figure 5, the overall performance plateaus once the number of HAPs exceeds twice that of the TDCs. Moreover, the average interval time (4ms) is much larger than the inference time (0.2ms) shown in Figure 6a. Therefore, the inference time of our algorithm is acceptable and still has advantages over the RL algorithm for a reasonable number of HAPs.

Figure 6b highlights the training cost. The RL algorithm requires substantially longer training to reach a usable policy, while all heuristic variants, including ρ, remain lightweight with negligible training requirements. This makes the proposed approach far more practical for deployment in large-scale or frequently changing topologies.

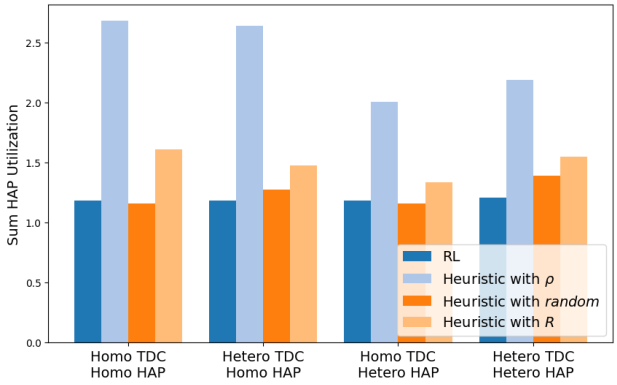


Fig. 7: Sum HAP utilization

*4) **Server-Level Heterogeneity**:* By server-level heterogeneity, we refer to variations in the effective number of servers available at each TDC and HAP, since in practice not all servers may be usable (e.g., some may be damaged, reserved for storage, or allocated to other functions). As a homogeneous baseline, we first consider a system with four TDCs, each equipped with 500 servers, and six HAPs, each with 30 servers. We then introduce heterogeneity by setting the servers in the TDCs to [100, 200, 500, 1000] and in the HAPs to [40, 40, 30, 30, 20, 20]. Figures 7 report the performance under these conditions.

The heuristic algorithm with the ρ indicator consistently achieves the highest HAP utilization while maintaining a low drop ratio, demonstrating robustness across both homogeneous and heterogeneous infrastructures. In contrast, the RL-based baseline shows poor convergence in the heterogeneous setting, resulting in significantly lower utilization and higher drop ratios. This highlights the difficulty of learning stable policies when server capacities vary widely across nodes. Note that when heterogeneous TDCs are considered along with heterogeneous HAPs, the performance rise a little. The reason is that now the HAP with more servers can be assigned by a TDC with more servers, which can balance the heterogeneity of server numbers of HAPs.

VII. Conclusion

This work has investigated efficient task-aware offloading algorithms for DC-HAP systems to reduce energy consumption. We derive a sampling-based heuristic algorithm combining our derived preference indicator. To compare, we construct an RL-based model, as well as a random and a throughput-based indicator for our sampling decision-making architecture, as baselines. The simulation results reveal that our heuristic algorithm with the ρ indicator consistently got the best performance, varying the utilization level of TDCs and topologies of the system, as well as the server-level heterogeneous situation. From the efficiency aspect, our heuristic algorithm got a lower time delay for decision making, while extremely low resource is needed for training. The outstanding performances make our heuristic algorithm suitable for the real-world deployment of the DC-HAP system to achieve next-generation green computing propagation.

REFERENCES

[1] W. Li, J. Liu, S. Wang, T. Zhang, S. Zou, J. Hu, W. Jiang, and J. Huang, "Survey on traffic management in data center network: from link layer to application layer," *IEEE Access*, vol. 9, pp. 38 427–38 456, 2021.

[2] Y. Li, Y. Wen, D. Tao, and K. Guan, "Transforming cooling optimization for green data center via deep reinforcement learning," *IEEE transactions on cybernetics*, vol. 50, no. 5, pp. 2002–2013, 2019.

[3] W. He, Q. Xu, S. Liu, T. Wang, F. Wang, X. Wu, Y. Wang, and H. Li, "Analysis on data center power supply system based on multiple renewable power configurations and multi-objective optimization," *Renewable Energy*, vol. 222, p. 119865, 2024.

[4] O. Amin, S. Dang, A. M. Abdelhady, G. Ma, J. Ye, M.-S. Alouini, and B. Shihada, "Beyond the Wi-Fi era," *Frontiers in Communications and Networks*, vol. 5, p. 1486488, 2024.

[5] Y. Chen, C. Lin, J. Huang, X. Xiang, and X. Shen, "Energy efficient scheduling and management for large-scale services computing systems," *IEEE Transactions on Services Computing*, vol. 10, no. 2, pp. 217–230, 2015.

[6] G. Premsankar and B. Ghaddar, "Energy-efficient service placement for latency-sensitive applications in edge computing," *IEEE internet of things journal*, vol. 9, no. 18, pp. 17 926–17 937, 2022.

[7] L. Gu, W. Zhang, Z. Wang, D. Zeng, and H. Jin, "Service management and energy scheduling toward low-carbon edge computing," *IEEE Transactions on Sustainable Computing*, vol. 8, no. 1, pp. 109–119, 2022.

[8] Y. Xu, H. Xu, X. Chen, H. Zhang, B. Chen, and Z. Han, "Blockchain-based ar offloading in uav-enabled mec networks: A trade-off between energy consumption and rendering latency," *IEEE Transactions on Vehicular Technology*, 2025.

[9] W. Abderrahim, O. Amin, and B. Shihada, "How to leverage high altitude platforms in green computing?" *IEEE Communications Magazine*, vol. 61, no. 7, pp. 134–140, 2023.

[10] K. Mershad, H. Dahrouj, H. Sarieddeen, B. Shihada, T. Al-Naffouri, and M.-S. Alouini, "Cloud-enabled high-altitude platform systems: Challenges and opportunities," *Frontiers in Communications and Networks*, vol. 2, p. 716265, 2021.

[11] W. Abderrahim, O. Amin, and B. Shihada, "Data center-enabled high altitude platforms: A green computing alternative," *IEEE Transactions on Mobile Computing*, 2023.

[12] A. Hammoud, A. Mourad, H. Otrok, O. A. Wahab, and H. Harmanani, "Cloud federation formation using genetic and evolutionary game theoretical models," *Future Generation Computer Systems*, vol. 104, pp. 92–104, 2020.

[13] J. Lu, O. Amin, and B. Shihada, "Task-aware offloading for cloud computing in sky-based aerial data centers," *TechRxiv*, 2025, preprint. [Online]. Available: https://doi.org/10.36227/techrxiv.175615804.41945749/v1

[14] O. Abbasi, A. Yadav, H. Yanikomeroglu, N.-D. Đào, G. Senarath, and P. Zhu, "Haps for 6g networks: Potential use cases, open challenges, and possible solutions," *IEEE Wireless Communications*, vol. 31, no. 3, pp. 324–331, 2024.

[15] Q. Ren, O. Abbasi, G. K. Kurt, H. Yanikomeroglu, and J. Chen, "Caching and computation offloading in high altitude platform station (haps) assisted intelligent transportation systems," *IEEE Transactions on Wireless Communications*, vol. 21, no. 11, pp. 9010–9024, 2022.

[16] J. Prodanov, B. Bertalanič, C. Fortuna, S.-K. Chou, M. B. Jurič, R. Sanchez-Iborra, and J. Hribar, "Multi-agent reinforcement learning-based in-place scaling engine for edge-cloud systems," in *2025 IEEE 18th International Conference on Cloud Computing (CLOUD)*. IEEE, 2025, pp. 32–42.

[17] T. Bao, X. Li, Y. Zhao, M. Lian, Y. Jiang, and J. Zhang, "Joint optimization of dnn model partitioning and slice delivery for distributed edge-cloud inference over optical networks," *Journal of Optical Communications and Networking*, vol. 17, no. 11, pp. 1047–1058, 2025.

[18] S. Pedratscher, T. Fahringer, and J. Aznar-Poveda, "Dynamic workflow scheduling in the edge-cloud continuum: Optimizing runtimes under budget constraints," in *2024 IEEE 17th International Conference on Cloud Computing (CLOUD)*. IEEE, 2024, pp. 69–80.

[19] T. Ekaireb, L. Brand, N. Avaraddy, M. Mock, C. Krintz, and R. Wolski, "Distributed dataflow across the edge-cloud continuum," in *2024 IEEE 17th International Conference on Cloud Computing (CLOUD)*. IEEE, 2024, pp. 316–327.

[20] N. Akbari, A. N. Toosi, J. Grundy, H. Khalajzadeh, M. S. Aslanpour, and S. Ilager, "Icontinuum: An emulation toolkit for intent-based computing across the edge-to-cloud continuum," in *2024 IEEE 17th International Conference on Cloud Computing (CLOUD)*. IEEE, 2024, pp. 468–474.

[21] H. Zhang, S. Huang, M. Xu, D. Guo, X. Wang, X. Wang, V. C. Leung, and W. Wang, "Large-scale measurements and optimizations on latency in edge clouds," *IEEE Transactions on Cloud Computing*, 2024.

[22] L. Yang, A. Zhou, X. Ma, Y. Zhang, and S. Wang, "Reliability-aware task replication for mobile edge computing," *IEEE Internet of Things Journal*, vol. 11, no. 14, pp. 24 846–24 857, 2024.

[23] M. Gherari, M. Dieye, H. Elbiaze, Y. Ghamri-Doudane, and R. H. Glitho, "3c resource allocation for next-generation applications in an in-network computing-enabled edge-cloud continuum," in *GLOBECOM 2024-2024 IEEE Global Communications Conference*. IEEE, 2024, pp. 614–619.

[24] G. P. Mattia, A. Pietrabissa, and R. Beraldi, "A load balancing algorithm for equalising latency across fog or edge computing nodes," *IEEE Transactions on Services Computing*, vol. 16, no. 5, pp. 3129–3140, 2023.

[25] Y. Yang, Z. Zhou, and L. Meng, "Heterogeneous resources adaptive co-optimization in edge networks," in *2025 IEEE International Conference on Web Services (ICWS)*. IEEE, 2025, pp. 420–427.

[26] K. Toledo, P. G. Kannan, M. Malka, E. Lev-Ran, O. Ozeri, V. Bortnikov, Z. Nevo, and K. Barabash, "Clusterlink: Redefining application connectivity for the multi-cloud era," in *2025 IEEE 18th International Conference on Cloud Computing (CLOUD)*. IEEE, 2025, pp. 210–222.

[27] P. Shukla and V. Patil, "A comprehensive review of frameworks for achieving interoperability in multi-cloud environments," in *2023 Second International Conference on Informatics (ICI)*. IEEE, 2023, pp. 1–6.

[28] H. Mostafaei, G. Smaragdakis, T. Zinner, and A. Feldmann, "Delay-resistant geo-distributed analytics," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4734–4749, 2022.

[29] J. Zhang, G. Zhao, J. Wen, H. Xu, W. Fan, X. Xu, and J. Yao, "Card: Cost-efficient and availability-aware application deployment in geo-distributed clouds," in *2025 IEEE/ACM 33rd International Symposium on Quality of Service (IWQoS)*. IEEE, 2025, pp. 1–10.

[30] A. Chakraborty, A. Eswaran, P. Thorat, M. Verma, P. Gupta, and P. Jayachandran, "Intent-driven multi-engine observability dataflows for heterogeneous geo-distributed clouds," in *2024 IEEE 17th International Conference on Cloud Computing (CLOUD)*. IEEE, 2024, pp. 30–41.

[31] L. Qin, H. Lu, Y. Chen, B. Chong, and F. Wu, "Towards decentralized task offloading and resource allocation in user-centric mec," *IEEE Transactions on Mobile Computing*, 2024.

[32] H. Vijayaraghavan, J. von Mankowski, and W. Kellerer, "Computifi: Latency-optimized task offloading in multipath multihop lifi-wifi networks," *IEEE Open Journal of the Communications Society*, 2024.

[33] H. T. Friis, "A note on a simple transmission formula," *Proceedings of the IRE*, vol. 34, no. 5, pp. 254–256, 1946.

[34] R. I. Davis, S. Kollmann, V. Pollex, and F. Slomka, "Controller area network (can) schedulability analysis with fifo queues," in *2011 23rd Euromicro Conference on Real-Time Systems*. IEEE, 2011, pp. 45–56.

[35] J. Lu, O. Amin, and B. Shihada, "Distributed computational offloading across multiple haps and terrestrial data centers," in *IEEE Global Commun. Conf., Taipei, China*, Dec. 2025.

[36] S. Wang, B. Yang, Z. Yu, X. Cao, Y. Zhang, and C. Yuen, "Computation offloading for multi-server multi-access edge vehicular networks: A ddqn-based method," in *2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring)*. IEEE, 2024, pp. 1–5.

[37] C. Liu, H. Wang, M. Zhao, J. Liu, X. Zhao, and P. Yuan, "Dependency-aware online task offloading based on deep reinforcement learning for iov," *Journal of Cloud Computing*, vol. 13, no. 1, p. 136, 2024.

[38] I. Ullah and Y.-H. Han, "Optimizing vehicular edge computing: graph-based double-dqn approaches for intelligent task offloading," *The Journal of Supercomputing*, vol. 81, no. 1, p. 76, 2025.

[39] L. Qin, H. Lu, Y. Chen, B. Chong, and F. Wu, "Toward decentralized task offloading and resource allocation in user-centric mec," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 11 807–11 823, 2024.

[40] A. Katal, S. Dahiya, and T. Choudhury, "Energy efficiency in cloud computing data centers: a survey on software technologies," *Cluster Computing*, vol. 26, no. 3, pp. 1845–1875, 2023.

[41] H. Guo, J. Liu, and J. Lv, "Toward intelligent task offloading at the edge," *IEEE Network*, vol. 34, no. 2, pp. 128–134, 2019.