Task-Aware Offloading for Cloud Computing in Sky-Based Aerial Data Centers

Jichen Lu, Osama Amin, Basem Shihada

***Abstract*—Data Center-Enabled High Altitude Platforms (DC-HAPs) present a promising solution for reducing terrestrial data centers' energy consumption through natural cooling and solar power harvesting. This paper investigates task-aware computational offloading strategies to further minimize terrestrial data center energy consumption. We first transform the energy consumption minimization objective to computational utilization maximization for HAPs. By modeling both transmission and computation as queuing systems, we derive closed-form expressions for maximum achievable computational utilization under both latency-constrained and unconstrained scenarios for single-type tasks, revealing the critical relationship between task instruction lengths and bit lengths. For multi-type task offloading, we formulate an optimization problem and develop a heuristic sequential algorithm guided by theoretical insights, alongside a novel feasibility-preserving transformation that significantly enhances traditional optimization approaches. Simulations demonstrate that our heuristic algorithm maintains consistent performance regardless of problem complexity, outperforming conventional methods while requiring substantially less computational time. This work provides a foundation for practical DC-HAP deployment in next-generation green computing infrastructure.**

***Index Terms*—Data Center-enabled High Altitude Platform (DC-HAP), green computing network, non-terrestrial network, task-aware offloading, queue delay**

I. Introduction

DATA centers are crucial components of modern digital infrastructures, supporting essential services such as web search, cloud computing, and distributed storage [1]. The rapid expansion of the IT industry and the proliferation of diverse applications have led to a surge in demand for data center services, resulting in significantly increased energy consumption [2]. Statistical projections indicate that global data center energy consumption, which accounted for 1.3% in 2010, is expected to rise to 10% by 2050 [3]. Beyond these energy consumption concerns, addressing this challenge is crucial for enabling broader access to computing resources while minimizing environmental impact [4]. This situation calls for innovative solutions that reduce energy consumption and enable ubiquitous access through alternative platforms.

Conventional terrestrial data centers (TDCs) consume substantial amounts of energy, with 30% – 40% dedicated to cooling and 26% – 56% to computing operations. To address this energy efficiency challenge, researchers have proposed hosting data centers on High Altitude Platforms (HAPs) [5]–[7]. Operating in the stratosphere at approximately 20 kilometers altitude, where temperatures range from -50° to -15°, these platforms can leverage the naturally cold environment to reduce or eliminate cooling energy requirements. Additionally, HAPs can utilize their expansive surface area to accommodate large solar panels, harvesting renewable energy to power computing operations and meet other energy needs. This solution promises to reduce overall data center energy consumption while promoting sustainable computing practices [5], [7].

Data center-enabled HAPs (DC-HAPs) systems show particular promise for several critical applications in 6G and beyond wireless systems [8]. In urban areas, they can enhance the reliability and maintainability of TDCs and other computing systems. One investigated application is to use DC-HAPs to assist the intelligent transportation system with their computing and machine abilities, which achieves lower delay for users [9]. For disaster-stricken and remote areas, these systems can also guarantee the availability of computing and communication services. The large terrestrial footprint and line-of-sight communication advantages of HAPs, combined with their naturally energy-efficient computing capabilities due to stratospheric temperatures, make them well-suited for these use cases [5].

However, implementing DC-HAP systems presents several technical challenges. First, the payload limitations of HAPs necessitate careful consideration of server numbers and characteristics to maintain platform stability. Second, the link capacity between ground facilities and HAPs must be properly managed to avoid communication bottlenecks and potential service outages. Third, the heterogeneous nature of computational workloads with varying quality-of-service requirements complicates resource allocation and scheduling decisions. Moreover, extreme weather conditions in the stratosphere and the need for regular maintenance of both airborne and computing systems add additional operational complexities [5].

Recent research on DC-HAP systems has investigated deployment feasibility with two distinct tracks. The first explores DC-HAP as a green computing solution for TDC computational load offloading. In this context, Abderrahim *et al.* conducted a comprehensive analysis of DC-HAP energy efficiency under varying conditions, including latitude, seasonal changes, and the number of HAPs deployed [7]. In addition, their study provides design and performance insights into system efficiency. The second track focuses on utilizing HAP-based data centers primarily to serve mobile users. Following this direction, Ding *et al.* examined a hybrid architecture

The authors are with the Computer, Electrical and Mathematical Sciences and Engineering (CEMSE) Divison, King Abdullah University of Science and Technology (KAUST), Thuwal 23955, Makkah Prov., Saudi Arabia (e-mail: {jichen.lu osama.amin, basem.shihada}@kaust.edu.sa)

combining HAPs and satellites equipped with servers for processing ground users' computational tasks [10]. However, their focus on mobile services limited the analysis to single-task processing scenarios. The comprehensive analysis of offloading performance between TDC and DC-HAP, considering multiple task types and high arrival rates, remains unexplored.

To address these challenges, we propose a task-aware offloading framework considering both computational capacity and communication delay aspects of the DC-HAP system. The main contributions of this study are summarized as follows:

- We introduce the first framework to analyze non-terrestrial computation with multi-type tasks, considering energy consumption and delay constraints in DC-HAP systems.
- We develop a comprehensive analytical model that integrates computational and transmission processes for DC-HAP task offloading, from which we derive closed-form expressions for the maximum computational utilization per task type.
- We formulate an optimization problem for single and multi-type task offloading based on our analytical model, and
- We formulate an optimization problem for single and multi-type task offloading based on our analytical model, and develop an efficient heuristic algorithm alongside tailored adaptations of two classical methods for solving this optimization challenge.
- We conduct extensive numerical simulations using a realistic simulation environment based on real-world parameters. We also construct a Deep Reinforcement Learning (DRL) based task offloading agent as a baseline, as a competitive baseline. Experimental results demonstrate that our proposed heuristic algorithm outperforms existing methods across key performance metrics, including computational utilization, task drop ratio, and energy efficiency, validating the effectiveness of our analytical framework.

The remainder of this paper is organized as follows: Section III presents the system model from communication and computation aspects. Section IV develops the task-aware offloading framework for a single type. Section V extends the scenario to multiple types. Section VI presents performance evaluation results and discussion. Finally, Section VII concludes the paper and suggests future research directions. The abbreviations and notations used in the following sections are shown in TABLE I and TABLE II.

TABLE I: List of Abbreviations

Abbreviation	Description
DC-HAP	Data Center-enabled High Altitude Platform
TDC	Terrestrial Data Center
HAP	High Altitude Platform
SCV	Squared Coefficient of Variation
IPS	Instructions Per Second
FSPL	Free Space Path Loss
SNR	Signal-to-Noise Ratio
FIFO	First-In-First-Out
SQP	Sequential Quadratic Programming
DE	Differential Evolution
DRL	Deep Reinforcement Learning

TABLE II: List of Notations

Symbol	Description
N_{s}	Number of servers at HAP
μ	Computing capacity per server (IPS)
N	Number of task types
l_i	Instruction length of task type i (instructions)
b_i	Bit length of task type i (bits)
λ_i	Offloaded task rate of type i (tasks/s)
λ_i^{TDC}	Task rate processed at TDC (tasks/s)
$\lambda_i^{\mathrm{total}}$	Total arrival rate of type i (tasks/s)
ρ_{comp}	HAP computational utilization
E	TDC energy consumption
R	Channel capacity (bit/s)
B	Channel bandwidth (Hz)
γ	Average SNR
L_{HAP}	Distance between TDC and HAP (m)
P_{trans}	Transmit power (W)
N_0	Noise power
T	Total sojourn time (s)
T_{trans}	Transmission delay (s)
T_{comp}	Computation delay (s)
W_{trans}	Transmission waiting time (s)
S_{trans}	Transmission service time (s)
W_{comp}	Computation waiting time (s)
S_{comp}	Computation service time (s)
ρ_{trans}	Transmission resource utilization
t_{delay}	Maximum allowed latency (s)

II. Related Work

Several studies have explored the feasibility and energy efficiency of DC-HAP systems [5], [7], [8]. Abbasi *et al.* demonstrated several applications for the HAP as a computing node [8]. They emphasized the advantage of HAP computing capacity compared to conventional edge devices. Besides traditional computing, DC-HAP has also been proved as a promising and reliable platform for quantum computing recently [11]. Abderrahim *et al.* [7] first proposed the concept of DC-HAPs and conducted a comprehensive analysis of their energy efficiency under various conditions, including latitude and seasonal variations. They modeled the energy consumption of TDCs, which mainly comes from the computational energy consumed by servers and the cooling energy. They found that the cooling energy can be significantly reduced due to the naturally cold environment in the stratosphere. As for the computational energy, they modeled the energy harvesting and consumption of the HAP, and found that the harvested energy typically exceeds the total energy required by the servers on the HAP [7]. This finding suggests that the DC-HAP is a promising solution for reducing energy consumption in TDCs by offloading computational tasks to HAPs. However, their analysis mainly focuses on the energy aspect and the feasibility of DC-HAPs, without considering multiple task types and the delay constraints that may arise in practical scenarios.

To the best of our knowledge, no prior work has addressed task offloading optimization for DC-HAP systems. Nevertheless, related studies on task offloading for non-terrestrial platforms, including HAPs, satellites, and UAVs, provide relevant context, though the task density and workload heterogeneity in

these works are relatively low compared to our data center application. For instance, Ding *et al.* [10] study a satellite–aerial integrated edge computing network, where low Earth orbit (LEO) satellites and HAPs collaboratively provide computing services. They formulate a joint optimization problem of transmission power and computation resource allocation to minimize the overall task processing delay. To address the non-convex problem constructed, a decomposition-based algorithm is proposed, enabling cooperative task execution across multiple nodes. Their results show that properly balancing communication and computation resources significantly reduces latency compared to separate optimization strategies. However, they assume a partial offloading protocol, where tasks can be partitioned and processed at different nodes, which may not be applicable for all types of tasks. Moreover, the task size and variability are not well considered in their analysis, which is not sufficient for the practical data center scenarios where tasks can have diverse characteristics and larger sizes. In [12], the authors investigate a space–air–ground–sea integrated architecture, where multiple high-altitude platforms (HAPs) collaboratively assist computation offloading for users in maritime scenarios. The problem is formulated as a joint optimization of task offloading and resource allocation, leveraging a decoupling strategy and optimization for each sub-problem to improve system performance under heterogeneous communication and computing constraints. The proposed framework enhances service availability and reduces overall latency in large-scale integrated networks. However, this work adopts a single-type optimization framework, where the computational resources can be allocated to each task independently, which may not be efficient or practical for scenarios with a large number of incoming tasks. In contrast, our work considers indivisible task offloading with multiple task types and incorporates queueing-based delay modeling, which more accurately captures the operational characteristics of DC-HAP systems.

Beyond conventional optimization approaches, learning-based methods have also been applied to task offloading in aerial networks and edge computing scenarios [13]–[19]. For instance, Yadav *et al.* investigate task offloading in Internet-of-Medical-Things (IoMT) systems, where a lightweight adaptive learning-based algorithm is proposed to dynamically balance the tradeoff between energy consumption and latency under resource-constrained and time-varying environments [20]. Li *et al.* [14] propose a MAPPO-based framework for task offloading and resource allocation in HAP-assisted LEO satellite networks, where multiple agents cooperatively optimize offloading decisions under a potential game formulation to improve system efficiency. Similarly, Zhao *et al.* [15] investigate a UAV-assisted MEC system and formulate the task offloading problem as a multi-agent Markov decision process (MDP), where UAVs and users act as agents and learn offloading policies through deep reinforcement learning. Tang *et al.* [16] propose a distributed task offloading framework based on advanced DQN variants, where each mobile device learns its offloading policy by interacting with dynamic edge environments. In multi-

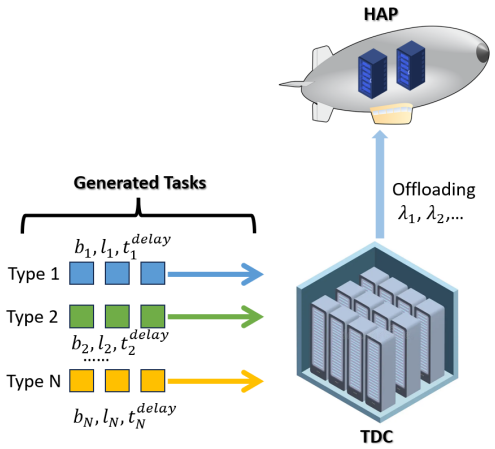


Fig. 1: Proposed task-aware offloading framework for DC-HAP systems.

UAV edge computing scenarios, Zakaryia *et al.* [17] and Tang *et al.* [18] further consider cooperative UAV systems, where DRL is employed to jointly optimize task offloading, resource allocation, and data collection, capturing the impact of UAV mobility and heterogeneous network schemes. In addition, Fan *et al.* [19] explore data offloading in smart city environments by leveraging learning-based approaches to exploit spatial location–functionality correlations. Despite the adaptability of DRL in handling system dynamics and high-dimensional decision spaces, these works typically adopt a task-wise or instantaneous system model, where a single task is assumed to be generated per time slot. Moreover, complex neural networks entail longer convergence times, longer execution times, and reduced flexibility under dynamic task variability. In contrast, our work considers a queue-based task offloading framework with heterogeneous task types, explicitly incorporating stochastic task arrivals under heavy task loads, which enables a more realistic characterization of delay, congestion, and long-term system performance in aerial edge computing systems.

III. System Modeling

In our DC-HAP system, TDCs can offload computational tasks to HAPs equipped with data center capabilities. Our study focuses on uplink transmission and HAP computation processes, assuming that the computational results on DC-HAP are relatively small and can be efficiently transmitted back to the source [21], [22]. Fig. 1 illustrates the comprehensive architecture of our proposed non-terrestrial task-aware offloading framework for DC-HAP systems. Throughout the rest of this section, we present the computation model, the wireless communication model, and the offloading delay model.

A. Computation Model

This subsection presents the computation model, defining the computational components, their processing capabilities, and task characteristics. We demonstrate how minimizing TDC

energy consumption can be reformulated as maximizing HAP computational utilization.

Consider a system with two types of computational components, TDCs and HAPs, both can undertake computational tasks. Assume that for each HAP, it has N_{s} servers, each with a computing capacity of μ instructions per second (IPS). There are N different types of tasks in the system, all of the tasks arrive at TDCs first, and can be processed at HAPs after offloading. For each task type $i \in 1, \ldots, N$, we define two key parameters: instruction length l_i (instructions/task), which indicates the number of instructions needed for processing the task; and bit length b_i (bits/task), which represents the data size required for transmission. The total arrival rate for each task type is denoted as $\lambda_i^{\mathrm{total}} = \lambda_{\mathrm{TDC},i} + \lambda_i$, where $\lambda_{\mathrm{TDC},i}$ (tasks/s) is the processing rate at the TDC servers , and λ_i (tasks/s) is the offloaded task rate to the HAP.

Following [7], HAP computational utilization is defined as:

$$\rho_{\mathrm{comp}} = \frac{\sum_{i=1}^{N} \lambda_i l_i}{N_{\mathrm{s}} \mu}, \tag{1}$$

where the numerator represents the total instruction rate to be processed at the HAP servers, and the denominator indicates the total computational capacity of the DC-HAP. In the following discussion, the computational utilization represents the computational utilization of the HAP for simplicity.

The primary objective of deploying DC-HAPs is to reduce energy consumption in TDCs. According to [7], the energy consumption E of the TDC is proportional to the total arrival instruction numbers to the TDC:

$$E \propto \sum_{i=1}^{N} \lambda_i^{\mathrm{TDC}} l_i \tag{2}$$

From (1), we have $\sum_i^N \lambda_i l_i = \rho_{\mathrm{comp}} N_{\mathrm{s}} \mu$. By expressing λ_i in terms of λ_i^{TDC} and $\lambda_i^{\mathrm{total}}$, we can express the processed instruction numbers at the TDC as,

$$\sum_{i=1}^{N} \lambda_i^{\mathrm{TDC}} l_i = \sum_{i=1}^{N} \lambda_i^{\mathrm{total}} l_i - \rho_{\mathrm{comp}} N_{\mathrm{s}} \mu. \tag{3}$$

Therefore, the energy consumption is inversely proportional to the computational utilization. Although we omit the complex nonlinear relationship between the energy saved in the TDC and the workload of the HAP, the objective of maximizing HAP utilization is valid for our system optimization problem for two reasons: 1) the objective is proportional to the energy saved in the TDC, since more offloaded tasks leads to fewer tasks processed within the TDC, and lower energy consumption, even under a non-linear relationship; 2) the objective serves as a consistent and comparable metric across different algorithms. Thus, in the following discussion, we use HAP utilization as our system's performance metric.

Due to payload limitations, the harvested energy typically exceeds the total energy required by the servers on the HAP, as demonstrated in [7]. Consequently, the objective of minimizing energy consumption in the TDC servers can be reformulated as maximizing the computational utilization. Having established the computation model and our optimization objective, we then present the wireless communication model that governs the task offloading process between TDCs and DC-HAPs.

B. Wireless Communication Model

In our analysis, we consider the upper bound on the data rate of the TDC-to-HAP link, $R = R_{\mathrm{UB}}$, for the communication system to ensure analytical tractability. Following [23], an upper bound on the achievable data rate is expressed as:

$$R_{\mathrm{UB}} = B \log_2 (1 + \gamma). \tag{4}$$

Under the considered communication conditions, the SNR is given by [24]–[28],

$$\gamma = \left(\frac{c}{4\pi f_{\mathrm{c}} L_{\mathrm{HAP}}} \right)^2 \times \frac{P_{\mathrm{trans}} 10^{\left(\frac{G_{\mathrm{t}} + G_{\mathrm{r}} - A_{\mathrm{gas}} - A_{\mathrm{rain}} - NF - L_{\mathrm{point}} - L_{\mathrm{impl}}}{10} \right)}}{N_0} \tag{5}$$

where c is the speed of light, f_{c} is the carrier frequency; L_{HAP} is the distance between the TDC and the HAP; P_{trans} is the transmit power of the TDC; $G_{\mathrm{t}}, G_{\mathrm{r}}$ are the antenna gains in dBi for transmitter and receiver respectively; N_0 is the thermal noise power expressed as $N_0 = kT_{\mathrm{ant}}B$, with the Boltzmann's constant $k = 1.38e - 23$ and the temperature at the receiver antennas T_{ant} [29], while NF is the noise figure in dB; $L_{\mathrm{point}}, L_{\mathrm{impl}}$ are pointing loss and implementation losses in dB; $(A_{\mathrm{gas}} + A_{\mathrm{rain}})$ is the attenuation from the atmosphere in dB.

C. Offloading Delay Model

Tasks offloaded to HAPs experience delays from two sources: transmission delays during the offloading process and computational delays at HAP servers with limited processing capacity. High task offloading volume can lead to congestion in both stages. We employ queueing theory to analyze these delays, modeling the offloading process as a tandem queue with transmission and computation stages.

1) Single-type Tasks Offloading Delay: For single-type tasks, the total sojourn time (T) of an offloaded task is the sum of its transmission delay (T_{trans}) during offloading and computational delay (T_{comp}) at the HAP:

$$T = T_{\mathrm{trans}} + T_{\mathrm{comp}}, \tag{6}$$

where each component includes both waiting time (W) and service time (S):

$$T_{\mathrm{trans}} = W_{\mathrm{trans}} + S_{\mathrm{trans}}, \tag{7}$$

$$T_{\mathrm{comp}} = W_{\mathrm{comp}} + S_{\mathrm{comp}}. \tag{8}$$

For the transmission process, we adopt a First-in-First-out (FIFO) $M/G/1$ queueing model [30], where the first M denotes a Poisson task arrival process and the second G indicates a general distribution for service times.

For a given task type with bit length b (bit/task), arrival rate λ (task/sec), and channel capacity R (bit/sec), the transmission components are:

$$W_{\text{trans}} = \frac{\rho_{\text{trans}}}{2(1-\rho_{\text{trans}})}\left(1+c_{\text{s,trans}}^2\right)\frac{b}{R} \tag{9}$$

$$S_{\text{trans}} = \frac{b}{R} \tag{10}$$

where $c_{\text{s,trans}}^2$ is the square of the coefficient of variation (SCV) of the transmission service time; ρ_{trans} represents transmission resource utilization defined as:

$$\rho_{\text{trans}} = \lambda\frac{b}{R} < 1, \tag{11}$$

with this constraint preventing infinite waiting times.

The computational queue has multiple servers, which can be modeled as a general $GI/G/m$ queue. Based on [31], the average waiting time of a general $GI/G/m$ queue under heavy traffic can be approximated as:

$$W_{\text{comp}} = \left(\frac{c_{\text{a,comp}}^2 + c_{\text{s,comp}}^2}{2}\right) EW(M/M/N_{\text{s}}) \tag{12}$$

To ensure more efficient use of computational resources, rather than randomly distributing incoming tasks among N_{s} servers during the routing, we apply the join-the-shortest-workload (JSW) algorithm to minimize the waiting time for each task [32]. Each task has a specific resource demand (transmission or computation), from which the expected service time can be estimated. Thus, we can have the expected waiting time for each server as the summation of the expected service time of all tasks in the queue, and route the next task to the queue with the shortest expected waiting time. Since we apply the JSW strategy for routing among HAP servers, based on [33], for an $M/M/m$ queue, the advantage of JSW over static routing is shown to approach a factor of m under heavy traffic, which gives:

$$EW(M/M/N_{\text{s}}/JSW) \approx \frac{1}{N_{\text{s}}} EW(M/M/1) \tag{13}$$

Combining the closed-form expression for $M/M/1$ queue [34], we have:

$$W_{\text{comp}} = \left(\frac{c_{\text{a,comp}}^2 + c_{\text{s,comp}}^2}{2}\right)\left(\frac{1}{N_{\text{s}}}\right)\frac{\rho_{\text{comp}}}{(1-\rho_{\text{comp}})}\frac{l}{\mu} \tag{14}$$

For the wireless transmission channel, there is typically a buffer to store the incoming packets [35]. During the buffer processing, a series of operations are performed, such as bit error checking and retransmission due to errors. Due to the rain attenuation, gaseous attenuation, and pointing loss, the introduced errors vary on the timescale of each transmission slot and are approximately independent across slots. Therefore, assuming that the offloaded tasks are successfully executed in the HAP, the number of transmission time slots required for retransmission can be approximated by a geometric distribution [36]. Given that each transmission time slot is relatively short, we can approximate the service time of such a wireless communication channel as exponential. Based on that, we do the approximation setting as $c_{\text{s,trans}}^2 \approx 1.0$, which leads to a tractability simplification with the transmission stage as an $M/M/1$ queue.

For the computational process, since with $c_{\text{s,trans}}^2 \approx 1.0$, the departure queue from the transmission stage can be approximated as a departure queue from an $M/M/1$ queue, which is still a Poisson Process. The interarrival time of the departure queue remains an exponential distribution [37], [38], which gives $c_{\text{a,comp}}^2 \approx 1.0$. Thus, we have:

$$W_{\text{comp}} = \frac{\rho_{\text{comp}}}{2(1-\rho_{\text{comp}})}\left(1+c_{\text{s,comp}}^2\right)\frac{l}{\mu}\left(\frac{1}{N_s}\right) \tag{15}$$

where $c_{\text{s,comp}}^2$ is the computational service time SCV, which is set to be $c_{\text{s,comp}}^2 = 0.1$ considering the server capacity is relatively stable. We define the utilization of the HAP ρ_{comp} as:

$$\rho_{\text{comp}} = \lambda\frac{l}{N_{\text{s}}\mu} < 1, \tag{16}$$

where this constraint ensures HAP servers can process all offloaded tasks.

The total sojourn time is therefore:

$$\begin{aligned} T =& \frac{\lambda\frac{b}{R}}{2(1-\lambda\frac{b}{R})}\left(1+c_{\text{s,trans}}^2\right)\frac{b}{R} + \frac{b}{R} \\ &+ \frac{\lambda\frac{l}{N_{\text{s}}\mu}}{2(1-\lambda\frac{l}{N_{\text{s}}\mu})}\left(1+c_{\text{s,comp}}^2\right)\frac{l}{N_{\text{s}}\mu} + \frac{l}{\mu}. \end{aligned} \tag{17}$$

2) Multi-type Tasks Offloading Delay: For multiple task types, the system becomes more complex as the DC-HAP receives different types of tasks from one or multiple TDCs. For the i-th task type with instruction length l_i and bit length b_i, the sojourn time is:

$$T_i = T_{\text{trans},i} + T_{\text{comp},i} \tag{18}$$

where each component includes its corresponding W and S:

$$T_{\text{trans},i} = W_{\text{trans},i} + S_{\text{trans},i} \tag{19}$$

$$T_{\text{comp},i} = W_{\text{comp},i} + S_{\text{comp},i}. \tag{20}$$

We model the system using $GI/G/1$ FIFO queues for each task type, where GI represents independent generally distributed arrival times and G represents general service time distribution. The transmission and computation utilizations for each type i are defined as $\rho_{\text{trans},i} = \lambda_i b_i/R$ and $\rho_{\text{comp},i} = \lambda_i l_i/(N_{\text{s}}\mu)$, respectively.

Following [39], the $i-\text{th}$ task transmission waiting time is:

$$\begin{aligned} W_{\text{trans},i} \approx& \frac{\sum_{j=1}^{N}\lambda_j E[X_j^2] + \rho_{\text{trans},j}E[X_j](c_{a_j}^2 - 1)}{2(1-\rho_{\text{trans}})} \\ &+ \frac{1}{2}E[X_i](c_{a_i}^2 - 1), \end{aligned} \tag{21}$$

where X_i represents service time for task type i, $\rho_{\text{trans}} = \sum_{j=1}^{N}\rho_{\text{trans},j}$ is the total utilization, and $c_{a_i} = 1$ for Poisson arrivals. $E[X_j^2] = (1+c_{\text{s,trans},j}^2)(\frac{b_j}{R})^2$ is the second moment

of service time, where $c_{\mathrm{s,trans},j}$ is the SCV of transmission time for type j, and $E[X_j] = \frac{b_j}{R}$ is the expected service time. Substituting into (21):

$$W_{\mathrm{trans},i} \approx \frac{\sum_{j=1}^{N} \lambda_j \left(1 + c^2_{\mathrm{s,trans},j}\right) \left(\frac{b_j}{R}\right)^2}{2(1-\rho_{\mathrm{trans}})}. \tag{22}$$

The service time $S_{\mathrm{trans},i} = b_i/R$ is similar to (10).

The transmission resource utilization must satisfy:

$$\rho_{\mathrm{trans}} = \frac{\sum_{i=1}^{N} \lambda_i b_i}{R} < 1. \tag{23}$$

For the multi-type queue, considering that the data center system may have a large number of types, we adopt the common approximation that the superposition of many independent renewal type-specific queues remains Poisson [40]. Combining with Burke's theorem [37], the total departure process can be approximated as Poisson, leading to $c^2_{\mathrm{d,trans}} = c^2_{\mathrm{a,comp}} \approx 1.0$ for the superposition of multi-type queues as a tractability simplification. With $\rho_{\mathrm{comp}} = \sum_{j=1}^{N} \rho_{\mathrm{comp},j}$ and $c_{\mathrm{s,comp},j}$ as the SCV of computational service time for type j, we have the SCV for the computational service time of such a superposition queue as:

$$c^2_{\mathrm{s,comp}} = \frac{\sum_{i}^{N} \left(\frac{\lambda_i}{\sum_i^N \lambda_i}(c^2_{\mathrm{s,comp},i} + 1)(\frac{l_i}{\mu})^2\right)}{\left(\sum_i^N \frac{\lambda_i}{\sum_i^N \lambda_i}(\frac{l_i}{\mu})\right)^2} - 1 \tag{24}$$

where $c^2_{\mathrm{s,comp},i}$ is the SCV for the computational service time of type i. We also apply a similar approximation in the multi-type average waiting time expression. By substituting (24) and $c^2_{\mathrm{a,comp}} \approx 1.0$ into (14), we have:

$$W_{\mathrm{comp},i} \approx \frac{\sum_{j=1}^{N} \frac{\lambda_j}{N_{\mathrm{s}}} \left(1 + c^2_{\mathrm{s,comp},j}\right) \left(\frac{l_j}{\mu}\right)^2 \left(\frac{1}{N_{\mathrm{s}}}\right)}{2(1-\rho_{\mathrm{comp}})}, \tag{25}$$

$$S_{\mathrm{comp},i} = \frac{l_i}{\mu}. \tag{26}$$

The computational resource utilization must satisfy:

$$\rho_{\mathrm{comp}} = \frac{\sum_{i=1}^{N} \lambda_i l_i}{N_{\mathrm{s}}\mu} < 1. \tag{27}$$

IV. Single-Type Offloading Scenario

In this section, we analyze the DC-HAP system's resource utilization based on the delay model developed in the previous section, considering single-type tasks with a latency constraint. Our optimization approach aims to maximize the computational utilization of the HAP data center while satisfying all system constraints, including latency requirements. Specifically, we optimize offloading task rate (λ) and the offloaded task properties (l, b) to maximize the computation utilization (ρ_{comp}), which can help us to define the DC-HAP operational limits under various conditions. In this regard, we define our design optimization problem as follows:

$$\begin{aligned}
\max_{\lambda,l,b} \quad & \rho_{\mathrm{comp}} && \text{(28a)}\\
\text{subject to} \quad & \rho_{\mathrm{trans}} < 1, && \text{(28b)}\\
& \rho_{\mathrm{comp}} < 1, && \text{(28c)}\\
& T \le t_{\max,} && \text{(28d)}\\
& \lambda > 0, && \text{(28e)}\\
& l_{\min} \le l \le l_{\max}, && \text{(28f)}\\
& b_{\min} \le b \le b_{\max}, && \text{(28g)}
\end{aligned}$$

where the constraints (28b) and (28c) ensures the transmission and computation stability conditions, respectively. The latency constraint is captured by (28d), while the λ positive conditions and task properties bounds are defined by (28e), (28f) and (28g), respectively. We assume that the TDC has much more servers than the HAP, and the generated task rate is high, which gives $\lambda_i^{\mathrm{total}} \gg \lambda_i$ [5], [7]. To solve (28), we decompose it into three subproblems with relaxed conditions and use their solutions to find the optimized λ, l and b, as will be seen in the rest of this section. Note that for the Ka-band carrier frequency, rain fade is highly dynamic under moderate-to-heavy precipitation, which will drastically impact the offloading bounds. In the rest of this section, we assume a Clear-Sky to Light-Rain atmosphere condition with relatively static fading parameters.

A. Maximizing Utilization without Latency Constraint

We first consider a a simplified version of (28) that focuses on optimizing λ for given task parameters l and b and relax the latency constraint:

$$\begin{aligned}
\max_{\lambda} \quad & \rho_{\mathrm{comp}} && \text{(29a)}\\
\text{subject to} \quad & \lambda > 0 && \text{(29b)}\\
& \rho_{\mathrm{trans}} < 1 && \text{(29c)}\\
& \rho_{\mathrm{comp}} < 1. && \text{(29d)}
\end{aligned}$$

Since ρ_{comp} is linear in λ, the optimal λ will be the maximum value that satisfies constraints (29c) and (29d) that are equivalently to $\lambda < \frac{R}{b}$ and $\lambda < \frac{N_{\mathrm{s}}\mu}{l}$, respectively. Hence, the utilization is maximized at $\lambda_{\max} = \min\left\{\frac{R}{b}, \frac{N_{\mathrm{s}}\mu}{l}\right\} - \epsilon$, where ϵ is an infinitesimally small positive number to strictly satisfy the constraints. $\lambda_{\max}$ can be written in terms of the task's computational-to-transmission ratio $\frac{l}{b}$ and the system's computational-to-transmission capacity ratio $\frac{N_{\mathrm{s}}\mu}{R}$ as:

$$\lambda_{\max} = \begin{cases} \dfrac{N_{\mathrm{s}}\mu}{l} - \epsilon, & \dfrac{l}{b} \ge \dfrac{N_{\mathrm{s}}\mu}{R} \\ \dfrac{R}{b} - \epsilon, & \dfrac{l}{b} < \dfrac{N_{\mathrm{s}}\mu}{R} \end{cases}. \tag{30}$$

By substituting $\lambda_{\max}$ into (16), we obtain the maximum achievable computational utilization $\rho_{\text{comp}}^{\max}$:

$$\rho_{\text{comp}}^{\max} = \begin{cases} 1 - \frac{\epsilon l}{N_{\text{s}}\mu}, & \frac{l}{b} \geq \frac{N_{\text{s}}\mu}{R} \\ \frac{lR}{N_{\text{s}}\mu b}, & \frac{l}{b} < \frac{N_{\text{s}}\mu}{R} \end{cases}. \tag{31}$$

B. Impact of Latency Constraint

We now extend our analysis to determine the maximum achievable computational utilization while satisfying an additional end-to-end latency constraint t_{delay}. Hence, the optimization problem is defined as follows:

$$\max_{\lambda} \quad \rho_{\text{comp}} \tag{32a}$$
$$\text{subject to} \quad \lambda > 0 \tag{32b}$$
$$\rho_{\text{trans}} < 1 \tag{32c}$$
$$\rho_{\text{comp}} < 1, \tag{32d}$$
$$T \leq t_{\text{delay}}. \tag{32e}$$

The following theorem establishes both the feasibility condition, the maximum offloaded task rate and the maximum achievable computational utilization considering a latency constraint, with the proof shown in the appendix.

Theorem 1. *In a DC-HAP system with N_{s} servers at the HAP, each with a computing capacity of μ IPS, and a communication link between the terrestrial and HAP data centers with a transmission rate R bit/sec, a task with a bit length b, instruction length l, and a latency constraint t_{delay} can be offloaded to the HAP if $t_{\text{delay}} > \frac{b}{R} + \frac{l}{\mu}$. In this case, the maximum offloaded task rate, $\tilde{\lambda}_{\max}$, is given by:*

$$\tilde{\lambda}_{\max} = \frac{-B - \sqrt{B^2 - 4AC}}{2A} \tag{33}$$

which provides a maximum computational utilization as:

$$\rho_{\text{comp}}^{\max} = \frac{\lambda_{\max} l}{N_{\text{s}}\mu} \tag{34}$$

where A, B, and C are defined as

$$A = 2\left(t_{\text{delay}} - \frac{b}{R} - \frac{l}{\mu}\right)\frac{b}{R}\frac{l}{N_{\text{s}}\mu} + \left(1 + c_{\text{s,trans}}^2\right)\frac{b^2}{R^2}\frac{l}{N_{\text{s}}\mu} + \left(1 + c_{\text{s,comp}}^2\right)\frac{l^2}{N_{\text{s}}^2\mu^2}\frac{b}{R} \tag{35a}$$

$$B = 2\left(t_{\text{delay}} - \frac{b}{R} - \frac{l}{\mu}\right)\left(-\frac{b}{R} - \frac{l}{N_{\text{s}}\mu}\right) - \left(1 + c_{\text{s,trans}}^2\right)\frac{b^2}{R^2} - \left(1 + c_{\text{s,comp}}^2\right)\frac{l^2}{N_{\text{s}}^2\mu^2} \tag{35b}$$

$$C = 2\left(t_{\text{delay}} - \frac{b}{R} - \frac{l}{\mu}\right). \tag{35c}$$

C. Optimal Task Properties for Constrained Systems

Building on Theorem 1, we now address (28) to determine the optimal task properties. Since ρ_{comp} is strictly increasing with λ in (16), and $\tilde{\lambda}_{\max}$ from Theorem 1 inherently satisfies all operational constraints, i.e., transmission stability (28b), computation stability (28c), latency requirements (28d), and positivity (28e), we substitute λ with $\tilde{\lambda}_{\max}(b, l)$ to obtain the following equivalent problem:

$$\max_{b,l} \quad \rho_{\text{comp}} = \frac{\tilde{\lambda}_{\max}(b,l)l}{N_{\text{s}}\mu} \tag{36a}$$
$$\text{subject to} \quad l_{\min} \leq l \leq l_{\max} \tag{36b}$$
$$b_{\min} \leq b \leq b_{\max}. \tag{36c}$$

To solve (36), we characterize two fundamental properties of the maximum computational utilization function through the following Lemmas, with the proof shown in the appendix.

Lemma 1. *Under transmission stability, computation stability, and latency constraints, $\rho_{\text{comp}}^{\max}$ of a DC-HAP system decreases strictly with bit length b, i.e., $\frac{\partial \rho_{\text{comp}}^{\max}}{\partial b} < 0$.*

Lemma 2. *Under transmission stability, computation stability, and latency constraints, ρ_{comp} of a DC-HAP system is concave in l, i.e., $\frac{\partial^2 \rho_{\text{comp}}^{\max}}{\partial l^2} < 0$.*

With Lemma 1 and Lemma 2, we propose Theorem 2 through the process shown in the appendix.

Theorem 2. *In a DC-HAP system with a latency constraint t_{delay}, the optimal task properties b^* and l^* that maximizes the computational utilization ρ_{comp} are:*

$$b^* = b_{\min} \tag{37}$$

$$l^* = \begin{cases} \hat{l}, & \text{if } \hat{l} \in [l_{\min}, l_{\max}] \\ \arg\max_{l \in \{l_{\min}, l_{\max}\}} \rho_{\text{comp}}^{\max}(b_{\min}, l), & \text{otherwise} \end{cases} \tag{38}$$

where $\hat{l}$ is computed from $\frac{\partial \rho_{\text{comp}}^{\max}}{\partial l}|_{(b_{\min}, \hat{l})} = 0$.

V. Multi-type Offloading Scenario

In this section, we extend our analysis to multi-type task offloading within a TDC-HAP system. We determine the optimal offloading rate λ_i for each task type i to maximize ρ_{comp}. To this end, we assume a given set of task types with properties $(\mathbf{b}, \mathbf{l})$. While Theorem 2 established offloading performance limits for the single-type scenario, we now focus on optimizing λ_i rather than $(\mathbf{b}, \mathbf{l})$, which are typically determined in data centers by application requirements [41]–[43].

A. Problem Formulation

Consider a system with N task types, each offloaded at rate λ_i. We define our problem that maximizes ρ_{comp} while meeting

delay constraints $t_{\text{delay},i}$ and ensuring stable operations as:

$$\max_{\lambda_i} \quad \rho_{\text{comp}}\left(\lambda_1, \ldots, \lambda_N\right) \tag{39a}$$

$$\text{subject to} \quad \lambda_i \geq 0, \forall i, \tag{39b}$$

$$\rho_{\text{trans}}\left(\lambda_1, \ldots, \lambda_N\right) < 1 \tag{39c}$$

$$\rho_{\text{comp}}\left(\lambda_1, \ldots, \lambda_N\right) < 1, \tag{39d}$$

$$T_i\left(\lambda_1, \ldots, \lambda_N\right) \leq t_{\text{delay},i}, \forall i. \tag{39e}$$

The non-convex nature of this optimization problem precludes finding a global optimum, requiring suboptimal iterative solution algorithms. However, these methods frequently produce intermediate solutions that violate the constraints, leading to instability, performance degradation and wasted computational resources. To overcome these feasibility violations, we adopt a feasibility-preserving transformation [44]–[46] that ensures all variables inherently satisfy the utilization constraints.

Firstly, we let $u_i = \frac{b_i}{R}\lambda_i$ and $v_i = \frac{l_i}{N_s \mu}\lambda_i$ to decouple the constraints (39c) and (39d). To address the resulting dependency between u_i and v_i, we redefine λ_i as:

$$\lambda_i = \min\left(\frac{R u_i}{b_i}, \frac{\mu N_{\text{s}} v_i}{l_i}\right). \tag{40}$$

This transformation ensures λ_i respects both utilization constraints, allowing u_i and v_i to be optimized independently. The optimization problem becomes:

$$\max_{u_i, v_i} \quad \rho_{\text{comp}} = \sum_{i=1}^{N} v_i \tag{41a}$$

$$\text{subject to} \quad u_i \geq 0, \forall i \tag{41b}$$

$$v_i \geq 0, \forall i \tag{41c}$$

$$\sum_{i=1}^{N} u_i < 1 \tag{41d}$$

$$\sum_{i=1}^{N} v_i < 1 \tag{41e}$$

$$T_i\left(\min\left(\frac{R u_1}{b_1}, \frac{\mu N_{\text{s}} v_1}{l_1}\right), \ldots, \min\left(\frac{R u_N}{b_N}, \frac{\mu N_{\text{s}} v_N}{l_N}\right)\right) \leq t_{\text{delay},i}, \forall i. \tag{41f}$$

Secondly, to ensure feasibility-preserving, we implement Softmax to impose constrained summation. To satisfy the inequality conditions in (41d) and (41e), we add slack variables u_0 and v_0 to vectors $\mathbf{u}$ and $\mathbf{v}$, and apply the Softmax function:

$$u_i = \mathcal{S}(\mathbf{u}') = \frac{e^{u_i'}}{\sum_{j=0}^{N+1} e^{u_j'}}, i \in [0, ..., N] \tag{42a}$$

$$v_i = \mathcal{S}(\mathbf{v}') = \frac{e^{v_i'}}{\sum_{j=0}^{N+1} e^{v_j'}}, i \in [0, ..., N] \tag{42b}$$

As a result, we can reformulate (41) with design variables $v_i', u_i' \in \mathbb{R}^1, \forall i \in [0, ..., N]$ while inherently satisfying constraints (41b), (41c), (41d) and (41e):

$$\max_{v_i', u_i'} \rho_{\text{comp}} \tag{43a}$$

$$\text{s.t.} \quad T_i\left(\min\left(\frac{R}{b_1}\frac{e^{u_1'}}{\sum_{j=0}^{N} e^{u_j'}}, \frac{\mu N_s}{l_1}\frac{e^{v_1'}}{\sum_{j=0}^{N} e^{v_j'}}\right), \ldots, \min\left(\frac{R}{b_n}\frac{e^{u_n'}}{\sum_{j=0}^{N} e^{u_j'}}, \frac{\mu N_s}{l_n}\frac{e^{v_n'}}{\sum_{j=0}^{N} e^{v_j'}}\right)\right) \leq t_{\text{delay},i}, \tag{43b}$$

$$\forall i \in [1, ..., N]. \tag{43c}$$

After solving the problem, we can calculate λ_i of task $i \in [1, ..., N]$ from:

$$\lambda_i = \min\left(\frac{R}{b_i}\frac{e^{u_i'}}{\sum_{j=0}^{N} e^{u_j'}}, \frac{\mu N_s}{l_i}\frac{e^{v_i'}}{\sum_{j=0}^{N} e^{v_j'}}\right). \tag{44}$$

For the complete validation proof of the transformation, we have the following lemmas, which are proved in the appendix:

Lemma 3. *The transformed problem* (43) *always satisfies constraints* (41b)-(41e) *in problem* (41)*, for* $\forall u_i', v_i' \in \mathbb{R}$ *after the transformation* (42)*. Equivalently, it satisfies constraints* (39c)-(39b) *in problem* (39) *after transformation* (42) *and recovery* (44)*.*

Lemma 4. *The domain of the transformed problem* (43) *remains the same as the original problem* (39) *after transformation* (42) *and recovery* (44)*.*

Therefore, Equation (42)-(44) ensures the utilization constraints are always satisfied, and it is a valid feasibility-preserving transformation, which maintains the same domain.

In the rest of this section, we develop an efficient heuristic algorithm guided by Theorem1. For comparative evaluation, we implement tailored adaptations of two classical methods: Sequential Quadratic Programming (SQP), which offers computational efficiency through gradient descent; and Differential Evolution (DE), which explores a broader solution space to increase the probability of finding the global optimum.

B. Heuristic Sequential Algorithm

Given the non-linear, non-convex nature of our optimization problem (39), finding a global optimum using conventional methods is challenging. We therefore develop a heuristic sequential algorithm that builds upon the insights from Theorem 1 while addressing the multi-type scenario. This approach leverages the computational utilization analysis for individual task types to construct an effective composite solution

Our strategy involves sequentially adding task types to the offloading schedule, prioritizing those with higher computational utilization potential. For each task type i, we first determine its maximum achievable computation utilization

using (33) and (34) from Theorem 1. We then sort types in descending order of their computational utilization and progressively incorporate them into our solution.

A key innovation in our approach is the adaptive scaling mechanism we employ when adding each new task type. Rather than simply adding tasks at their individual optimal rates, we recognize that task interactions may require adjustments to previously allocated resources. For each newly considered task type, we define a scale factor $r_s \in [0, 1]$ that scales the previously selected task rates $\lambda_{\text{prev}} = [\lambda_1, \lambda_2, ..., \lambda_{n-1}]$. We then conduct a linear search with $n_{\text{step}} = 1000$ intervals to find the optimal balance between existing and new task types.

For a given scaling factor r_s, we can derive the optimal offloading rate λ_{n,r_s} for the newly added task type n as:

$$\lambda_{n,r_{\text{s}}} = \frac{-B' - \sqrt{B'^2 - 4A'C'}}{2A'} \tag{45}$$

The expression of the variables is similar to (35):

$$f(\lambda_{n,r_{\text{s}}}) = A'\lambda^2 + B'\lambda + C' \tag{46a}$$

$$A' = A \tag{46b}$$

$$\begin{aligned} B' = {} & 2\left(t_{\text{delay}} - \frac{b}{R} - \frac{l}{\mu}\right)\left(-\frac{b}{R}(1+z_4) - \frac{l}{N_{\text{s}}\mu}(1+z_2)\right) \\ & - \left(1 + c^2_{\text{s,trans}}\right)\frac{b^2}{R^2}(1+z_4) \\ & - \left(1 + c^2_{\text{s,comp}}\right)\frac{l^2}{N_{\text{s}}^2\mu^2}(1+z_2) + \frac{b}{R}z_3 + \frac{l}{N_{\text{s}}\mu}z_1 \end{aligned} \tag{46c}$$

$$\begin{aligned} C' = {} & 2\left(t_{\text{delay}} - \frac{b}{R} - \frac{l}{\mu}\right)(1+z_2)(1+z_4) - z_1(1+z_4) \\ & - z_3(1+z_2) \end{aligned} \tag{46d}$$

where the auxiliary variables z_1 through z_4 capture the influence of previously selected tasks:

$$z_1 = \sum_{i=1}^{n-1} r_{\text{s}}\lambda_i \frac{b_i^2}{R^2}\left(1 + c^2_{\text{s,trans}}\right) \tag{47a}$$

$$z_2 = -\sum_{i=1}^{n-1} r_{\text{s}}\lambda_i \frac{b_i}{R} \tag{47b}$$

$$z_1 = \sum_{i=1}^{n-1} r_{\text{s}}\lambda_i \frac{l_i^2}{N_{\text{s}}^2\mu^2}\left(1 + c^2_{\text{s,comp}}\right) \tag{47c}$$

$$z_2 = -\sum_{i=1}^{n-1} r_{\text{s}}\lambda_i \frac{l_i}{N_{\text{s}}\mu} \tag{47d}$$

An important consideration is that adding new task types increases system congestion, potentially violating delay constraints for previously added tasks. To address this, we implement a final rescaling step using *binary search* to find a rescale factor r_b that, when applied to all task rates $r_b[r_s\lambda_1, r_s\lambda_2, ..., \lambda_{n,r_s}]$, ensures all sojourn time constraints remain satisfied. The binary search stops when the left and right margin is less than $\epsilon = 1e-6$ or the iteration number is larger than $m_{\text{iter}} = 1000$. This guarantees the feasibility

Algorithm 1: Heuristic Sequential Algorithm

Input:
$R, \mu, N_s, c^2_{\text{s,trans}}, c^2_{\text{s,comp}}, \mathbf{b}, \mathbf{l}, \mathbf{t}_{\text{delay}}, n_{\text{step}}, m_{\text{iter}}, \epsilon$
Output: $\boldsymbol{\lambda}, \rho_{\text{comp}}$
Initialize selected set $S \leftarrow \emptyset$, $\boldsymbol{\lambda} \leftarrow \mathbf{0}$, $\rho_{\text{comp}} \leftarrow 0$
Compute $\lambda_{\max,i}$ and $\rho^{\max}_{\text{comp},i}$ using (33)–(34).
Sort types into list L.
for type $i \in L$ **do**
  if $S = \emptyset$ **then**
    $\boldsymbol{\lambda} \leftarrow \lambda_{\max,i}$; $\rho_{\text{comp}} \leftarrow \rho^{\max}_{\text{comp},i}$
    $S \leftarrow S \cup \{i\}$
  else
    Initialize $r_s \leftarrow 0$, best candidate $(\boldsymbol{\lambda}_{\text{prev}}, \lambda_i)$
    for $j = 1$ to n_{step} **do**
      $r'_s \leftarrow \frac{j}{n_{\text{step}}}$; $\boldsymbol{\lambda}'_{\text{prev}} \leftarrow r'_s\boldsymbol{\lambda}$
      Compute λ'_i using (45).
      Evaluate ρ'_{comp}.
      if $\rho'_{\text{comp}} \geq \rho_{\text{comp}}$ **then**
        $(r_s, \boldsymbol{\lambda}_{\text{prev}}, \lambda_i) \leftarrow (r'_s, \boldsymbol{\lambda'}_{\text{prev}}, \lambda'_i)$.
      end if
    end for
    repeat
      Binary search on $(\boldsymbol{\lambda}_{\text{prev}}, \lambda_i)$ to satisfy (39e).
    until margin < ϵ and iteration > m_{iter}
    Compute ρ''_{comp} using (1)
    if $\rho''_{\text{comp}} \geq \rho_{\text{comp}}$ **then**
      $\boldsymbol{\lambda} \leftarrow$ refined solution; $\rho_{\text{comp}} \leftarrow \rho'_{\text{comp}}$.
    end if
    $S \leftarrow S \cup \{i\}$
  end if
end for
return $\boldsymbol{\lambda}, \rho_{\text{comp}}$

of our composite solution while maximizing computational utilization. The complete procedure is detailed in Algorithm 1. The type-wise properties are provided via vector $\mathbf{b}$ for bit length, $\mathbf{l}$ for instruction length, and $\mathbf{t}_{\text{delay}}$ for delay constraint. Output λ is the vector of task rates for each type of task.

Complexity Analysis: The complexity of the algorithm is $O(Nn_{\text{step}} \log n_{\text{step}})$, where N is the number of task types and n_{step} is the number of linear search steps, which is a constant. The $\log n_{\text{step}}$ factor arises from the binary search used to determine the r_{b} in each linear search step. The n_{step} is a constant, which decide the search accuracy. Therefore, the complexity is linear with the number of types of tasks in the set, and we can control the accuracy of the search by adjusting the n_{step}.

C. Sequential Quadratic Programming Algorithm

Having transformed our original problem (39) into the form shown in (43), we ensure that hard utilization constraints are inherently satisfied. This transformation enables more effective application of gradient-based optimization methods. We im-

plement SQP, a powerful technique for constrained non-linear optimization problems [47], [48]. Note that our implementation adapts the Sequential Least Squares Programming (SLSQP) variant specifically to our transformed problem, due to its efficiency and robustness, with complexity analysis provided. While we reference the transformed problem formulation (43) in our discussion and algorithm presentation, we will compare results from both formulations in our experimental evaluation. It is important to note that since the problem is non-convex, SQP cannot guarantee global optimality but often produces high-quality local optima.

SQP operates by iteratively solving quadratic approximations of the original problem. To prevent excessive computation time, we impose a maximum iteration limit m_{iter}.

SQP starts by constructing the Lagrangian function for (43):

$$L = \rho_{\text{comp}} + k(\mathbf{T} - \mathbf{t}^{\text{delay}}) \tag{48}$$

where k represents the Lagrange multiplier associated with the delay constraint. At each iteration, SQP constructs and solves a quadratic programming subproblem:

$$\begin{aligned} &\min_{p} \ \frac{1}{2} p^T H p + g^T p \\ &\text{s.t.} \ \ (\mathbf{T} - \mathbf{t}^{\text{delay}}) + \nabla (\mathbf{T} - \mathbf{t}^{\text{delay}})^T p \leq 0 \end{aligned} \tag{49}$$

where p is the search direction, H is the Hessian matrix of the Lagrangian function, and g is the gradient of the Lagrangian function. This subproblem linearizes the constraints while approximating the objective function with a quadratic model. The detailed procedure is listed in the Algorithm 2. The SQP algorithm stops when the search direction norm $||p|| < \epsilon = 1e-6$ or the iteration number is larger than $m_{\text{iter}} = 1000$. The type-wise properties are provided via vector $\mathbf{b}$ for bit length, $\mathbf{l}$ for instruction length, and $\mathbf{t}_{\text{delay}}$ for delay constraint. Output λ is the vector of task rates for each type of task.

Complexity Analysis: The complexity of the SQP algorithm is $O(m_{\text{iter}} N^3)$, where m_{iter} is the maximum iteration limit, and N is the number of types of tasks in the set. The $O(N^3)$ is due to the Hessian matrix calculation, which is a cubic complexity with the number of variables. The m_{iter} is a constant, which decides the maximum iteration limit. Therefore, the complexity is cubic with the number of types of tasks in the set. However, the real time spent can vary due to the convergence speed of the algorithm, which is hard to predict since the problem is non-convex and non-linear.

Algorithm 2: Sequential Quadratic Programming

Input: $R, \mu, N_s, c^2_{\text{s,trans}}, c^2_{\text{s,comp}}, \mathbf{b}, \mathbf{l}, \mathbf{t}_{\text{delay}}, m_{\text{iter}}, \epsilon$
Output: $\boldsymbol{\lambda}, \rho_{\text{comp}}$
Initialize $(\mathbf{u}', \mathbf{v}')$, Lagrange multiplier k; $c_{\text{iter}} \leftarrow 0$.
repeat
  Transform $\mathbf{u} = \text{Softmax}(\mathbf{u}')$, $\mathbf{v} = \text{Softmax}(\mathbf{v}')$.
  $\lambda_i = \min\left(\frac{u_i R}{b_i}, \frac{v_i \mu N_s}{l_i}\right)$
  Identify active branch for optimization.
  Construct Lagrangian $\mathcal{L}$ as (48).
  Compute gradient $\nabla\mathcal{L}$ and Hessian $\nabla^2\mathcal{L}$.
  Solve subproblem (49) to obtain p.
  Perform line search to get step size α.
  Update $(\mathbf{u}', \mathbf{v}') \leftarrow (\mathbf{u}', \mathbf{v}') + \alpha p$.
  Update Lagrange multiplier k.
  $c_{\text{iter}} \leftarrow c_{\text{iter}} + 1$
  if $\|p\| < \epsilon$ **then**
    break
  end if
until $c_{\text{iter}} > m_{\text{iter}}$
Compute $\boldsymbol{\lambda}$ using final $(\mathbf{u}, \mathbf{v})$ with (44), check (43c).
return $\boldsymbol{\lambda}, \rho_{\text{comp}}$

D. Differential Evolution Algorithm

While gradient-based methods like SQP can effectively find local optima, the non-linear and non-convex nature of our optimization problem motivates the exploration of alternative approaches capable of searching more broadly for global solutions. DE represents a powerful population-based metaheuristic particularly well-suited for such challenging scenarios [49], [50]. Based on evolutionary principles, DE efficiently explores broad solution spaces to find close-to-optimal solutions. Our implementation adapts the algorithm with a custom candidate selection mechanism specifically designed for our problem formulation, accompanied by a comprehensive time complexity analysis.

Similar to the previously discussed hard utilization constraints, the evolutionary algorithm may generate initial population samples that fall outside the feasible region. Therefore, we compare both problem formulations in our implementation and use (43) as the reference formulation in the following discussion and algorithm pseudo-code.

Our DE implementation begins by generating an initial population of N_{pop} random candidate solutions. To manage computational resources, we limit the evolutionary process to a maximum of m_{iter} generations. The parameters F and r_{c} control the differential weight and crossover rate, respectively. Given our constrained optimization context, we implement a modified selection mechanism that prioritizes feasibility while promoting improved objective values:

- When comparing two feasible solutions, select the one with the higher computational utilization.
- If one solution is feasible and the other is infeasible, select the feasible one.
- If both solutions are infeasible, select the one with the smaller constraint violation.

After that, we randomly select two individuals to generate a trial vector $(\mathbf{u}_t, \mathbf{v}_t)$:

$$\mathbf{u}_t = \mathbf{u}_{\text{best}} + F(\mathbf{u}_b - \mathbf{u}_c) \tag{50a}$$

$$\mathbf{v}_t = \mathbf{v}_{\text{best}} + F(\mathbf{v}_b - \mathbf{v}_c) \tag{50b}$$

Then, perform the crossover to generate the new individual $(\mathbf{u}_{\text{new}}, \mathbf{v}_{\text{new}})$ for each individual:

$$u_{\text{new},i} = \begin{cases} u_{t,i} & \text{if rand}(0,1) \leq r_{\text{c}} \text{ or} \\ & i = \text{rand}(1, N) \\ u_i & \text{otherwise} \end{cases} \tag{51}$$

$$v_{\text{new},i} = \begin{cases} v_{t,i} & \text{if rand}(0,1) \leq r_{\text{c}} \text{ or} \\ & i = \text{rand}(1, N) \\ v_i & \text{otherwise} \end{cases} \tag{52}$$

Finally, replace the old individuals if the new one outperforms the old one based on the aforementioned rules. The detailed procedure is shown in the Algorithm 3. We also apply the SciPy library in Python to implement the Differential Evolution algorithm for our problem. The DE algorithm stops when the best individual between generations has improvement $\Delta\rho_{\text{comp}} < \epsilon = 1e-6$ or the iteration number is larger than $m_{\text{iter}} = 1000$. The type-wise properties are provided via vector $\mathbf{b}$ for bit length, $\mathbf{l}$ for instruction length, and $\mathbf{t}_{\text{delay}}$ for delay constraint. Output λ is the vector of task rates for each type of task.

Complexity Analysis: The complexity of the DE algorithm is $O(m_{\text{iter}}(N_{\text{pop}}N + N_{\text{pop}}^2))$, where m_{iter} is the maximum iteration/generation limit, N_{pop} is the population size, and N is the number of types of tasks in the set. The $O(N_{\text{pop}}N)$ is due to the crossover operation, and the $O(N_{\text{pop}}^2)$ is due to the selection of the best individual. The m_{iter} is a constant, which decides the maximum iteration limit. The real time spent can vary due to the convergence speed of the algorithm, which is hard to predict since the problem is non-convex and non-linear.

Algorithm 3: Differential Evolution Algorithm

Input:
$R, \mu, N_s, c^2_{\text{s,trans}}, c^2_{\text{s,comp}}, \mathbf{b}, \mathbf{l}, \mathbf{t}_{\text{delay}}, N_{\text{pop}}, m_{\text{iter}}, \epsilon$
Output: $\boldsymbol{\lambda}, \rho_{\text{comp}}$
Initialize population $\{(\mathbf{u}', \mathbf{v}')\}_1^{N_{\text{pop}}}$; $c_{\text{iter}} \leftarrow 0$
repeat
  for each individual $(\mathbf{u}', \mathbf{v}')$ **do**
    Transform $\mathbf{u} = \text{Softmax}(\mathbf{u}')$, $\mathbf{v} = \text{Softmax}(\mathbf{v}')$ as (42).
    Calculate $\lambda_i = \min\left(\frac{u_i R}{b_i}, \frac{v_i \mu N_s}{l_i}\right)$.
    Evaluate ρ_{comp} and delay constraint.
  end for
  Select $(\mathbf{u}_{\text{best}}, \mathbf{v}_{\text{best}})$ as rules in V-D.
  for each individual $(\mathbf{u}, \mathbf{v})$ **do**
    Select random $(\mathbf{u}_b, \mathbf{v}_b), (\mathbf{u}_c, \mathbf{v}_c)$.
    Generate $(\mathbf{u}_t, \mathbf{v}_t)$ with $(\mathbf{u}_b, \mathbf{v}_b), (\mathbf{u}_c, \mathbf{v}_c)$ as (50).
    Crossover $((\mathbf{u}, \mathbf{v}), (\mathbf{u}_t, \mathbf{v}_t))$ to $(\mathbf{u}_{\text{new}}, \mathbf{v}_{\text{new}})$ as in (51)-(52).
    Evaluate new individuals.
    Replace $(\mathbf{u}, \mathbf{v})$ with $(\mathbf{u}_{\text{new}}, \mathbf{v}_{\text{new}})$ if better.
  end for
  $c_{\text{iter}} \leftarrow c_{\text{iter}} + 1$
  if best individual improvement $\Delta\rho_{\text{comp}} < \epsilon$ **then**
    break
  end if
until $c_{\text{iter}} > m_{\text{iter}}$
Compute $\boldsymbol{\lambda}$ using best individual with (44), check (43c).
return $\boldsymbol{\lambda}, \rho_{\text{comp}}$

VI. Simulation Results

We consider a DC-HAP system with one TDC and one HAP positioned 20 km directly above the TDC ($L_{\text{HAP}} = 20$km). The HAP is equipped with $N_{\text{s}} = 20$ identical servers, each with computational capacity $\mu = 580$MIPS (million instructions per second) [7]. The link budget is shown in TABLE III. Following [51], we simulate tasks with data sizes ranging from 300KB to 800KB and computational requirements from 100 to 1000 MegaCycles, with an average cycles-to-instructions ratio of 3.7. We set the latency constraint to 500ms for all subsequent simulations unless otherwise stated. The overall simulation settings are shown in TABLE IV.

TABLE III: Link Budget Table

Parameter	Symbol	Value / Range
Carrier frequency	f_{c}	31 GHz
Tx power (TDC)	P_{trans}	5 W
Rx temperature	T_{ant}	216.5 K [52]
Bandwidth	B	50 MHz
Tx antenna gain	G_{t}	20 dBi
Rx antenna gain	G_{r}	20 dBi
Noise figure	NF	2 dB
Pointing loss	L_{point}	1 dB
Implementation loss	L_{impl}	2 dB
Gaseous attenuation	A_{gas}	0.5 dB
Rain attenuation	A_{rain}	3 dB

TABLE IV: System Parameters

Parameter	Symbol	Value / Range
HAP altitude	L_{HAP}	20 km
Servers per HAP	N_{s}	20
Server capacity	μ	580 MIPS [7]
Task data size	b	300–800 KB [51]
Task compute load	l	100–1000 MCycles [51]
Cycles-to-instructions ratio	CPI	3.7 [51]
Latency constraint	t_{delay}	500 ms

A. Single-type Offloading Scenario

We first investigate the performance of $\rho_{\text{comp}}^{\max}$ without latency constraints, as defined in (31) and illustrated in Fig. 2. Here, we present heatmaps showing the performance of the system under different (b, l) type task queues. The ranges of b and l follow settings in TABLE IV. The color bar indicates the metric values. The results reveal that the relationship between instruction length and bit length follows a specific proportional pattern for achieving high computational utilization, with the optimal ratio following a linear boundary. The $\rho_{\text{comp}}^{\max} = 1$ boundary represents the system's operational limit, above which the computational resources become fully saturated. Data center operators can use this boundary as

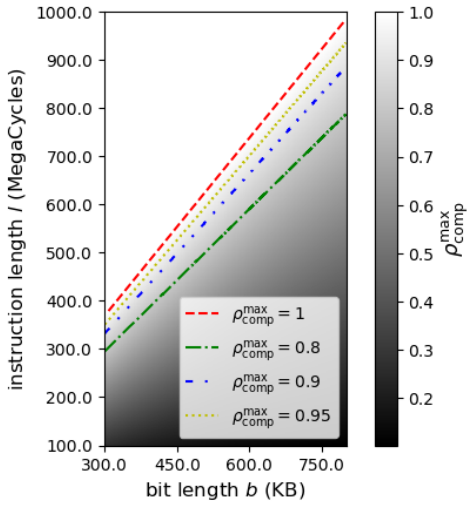


Fig. 2: $\rho^{\max}_{\text{comp}}$ for tasks without latency constraints.

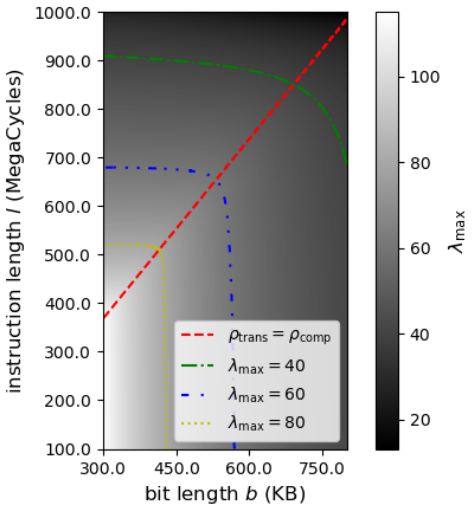


Fig. 3: $\lambda_{\max}$ for tasks under latency constraint of 500ms.

a guideline for identifying ideal task types for offloading when energy reduction is the primary goal. This finding has significant practical implications: when latency is not critical (e.g., for batch processing tasks), selecting tasks whose l and b values fall within the optimal region can ensure maximum computational utilization that maximizes the energy efficiency benefits of DC-HAPs.

When considering the practical 500ms latency constraint, Fig. 3 shows $\lambda_{\max}$ for different l and b values. The results demonstrate that smaller l and b values generally enable higher $\lambda_{\max}$, as expected due to reduced transmission and processing times per task. Particularly noteworthy is the red line indicating where $\rho_{\text{trans}} = \rho_{\text{comp}}$. This boundary creates two distinct operational regions: below the line (transmission-limited region), reducing bit length yields the most significant improvements in offloading capacity; above the line (computation-limited region), reducing instruction length is more effective. This insight provides system operators with clear guidance on which task characteristics to prioritize for optimization based on their workload properties.

Fig. 4 presents $\rho^{\max}_{\text{comp}}$ under the latency constraint, combining the effects of l and b on λ and hence $\rho^{\max}_{\text{comp}}$. Unlike the unconstrained scenario, the boundaries for high utilization form curved regions due to the complex interplay between two competing factors: 1) a larger b or l leads to a larger waiting time and limits the $\lambda_{\max}$; 2) a larger $\lambda_{\max}$ does not necessarily yield higher $\rho^{\max}_{\text{comp}}$ without sufficient instruction length l. This fundamental trade-off creates the curve-like boundaries in Fig. 4 traces the optimal instruction length for each bit length, providing system operators with a practical guide for task prioritization to maximize energy savings.

B. Multi-type Offloading Scenario

We now evaluate multi-type task scenarios using our Heuristic Sequential Algorithm compared against SQP and DE algorithms, the later two are solved using both Problem (39) and its

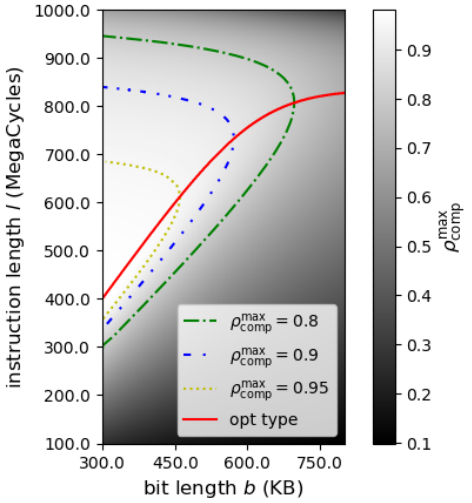


Fig. 4: $\rho^{\max}_{\text{comp}}$ for tasks under latency constraint of 500ms.

transformed version Problem (43). We show the performance of the algorithms under different numbers of task types. To evaluate the generalization ability of the algorithms across varying numbers of task types, in each set of experiments, we randomly sample 10 task types with property pairs (b, l) within the range shown in TABLE IV. Tasks are ordered by their $\rho^{\max}_{\text{comp}}$ values (calculated using Theorem 1) from high to low and added to the set one by one. Then let each algorithm run on these cumulative sets and record the performances. To obtain more general results and mitigate the effect of randomness, the average performance is evaluated across 1000 random experimental runs, with results shown in Fig. 5.

Our heuristic sequential algorithm and the DE algorithm consistently achieve the highest average $\rho^{\max}_{\text{comp}}$. SQP shows significant performance degradation as N increases, while DE exhibits greater robustness. This difference stems from the non-convex nature of the problem: gradient-based algorithms

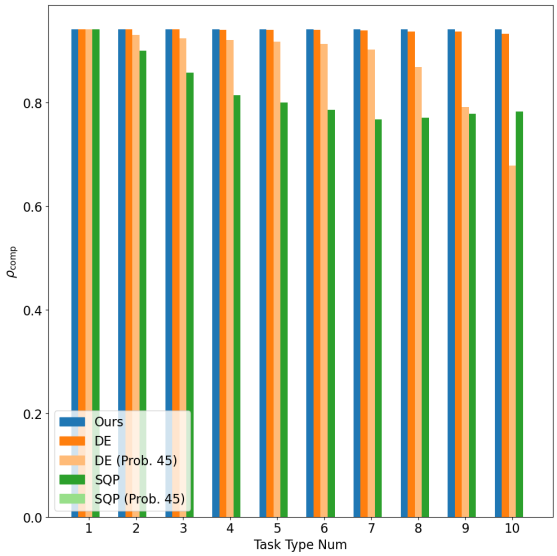


Fig. 5: Average $\rho_{\text{comp}}^{\max}$ for different algorithms.

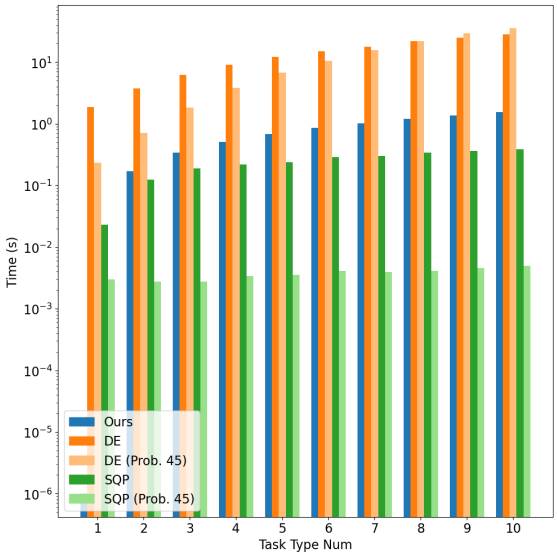


Fig. 6: Average runtime for different algorithms.

like SQP are prone to local minima entrapment, while DE's random initialization and mutation operations better navigate the complex search space. Notably, the heuristic sequential algorithm maintains consistent performance regardless of problem size, indicating that the theoretically-identified best type already achieves near-optimal computational utilization, with additional types offering minimal benefit.

Our feasibility-preserving transformation substantially improves algorithm reliability. Without it, DE experiences performance degradation with increasing dimensions, while SQP frequently fails to generate valid solutions by entering regions with incompatible constraints (utilizations exceeding 1 and waiting times becoming negative). This demonstrates the essential nature of our transformation approach for practical implementation.

For computational efficiency (Fig. 6), SQP requires the least time, followed by our heuristic algorithm, $i-\text{th}$ DE, taking the longest due to its population size and mutation operations. For practical implementation, we could further optimize our heuristic by eliminating the sequential searching step in Algorithm 1, effectively using only the solution from $N = 1$ when rapid computation is required. However, with the original formulation, DE experiences a dramatically increasing runtime as problem complexity grows due to generating infeasible solutions.

In conclusion, our heuristic algorithm provides the optimal balance between solution quality and computational efficiency for TDC-HAP task offloading, while our feasibility-preserving transformation significantly enhances both SQP and DE performance in complex multi-type scenarios. This demonstrates that theoretical insights can guide practical algorithm development more effectively than increasing algorithmic complexity alone.

C. Link Budget Sensitivity Analysis

Here, we analyze the metric $\rho_{\text{comp}}^{\max}$ sensitivity to the link budget components under a single-type scenario in Fig. 7. To conserve space, we use box plots instead of heatmaps to present the statistical performance across the task type range shown in TABLE IV. Note that the performance of the system is ensured under good weather conditions and hardware capacity. When the wireless communication channel experiences adverse conditions, due to weather or antenna settings, the performance drops significantly. Thus, the channel condition represents a key bottleneck of the DC-HAP system.

Besides the results for single types of tasks, we evaluate the sensitivity of our algorithms to changes in transmission conditions. Since the link budget comprises numerous variables that jointly determine the overall transmission condition, performing a single-factor sensitivity analysis for each variable is impractical. Thus, we directly show the algorithms' performance under a series of transmission throughputs, which directly captures the aggregate influence of the link budget parameters on system performance. Here, we present the performance for 10 different task types at the TDC. The result is shown in Fig. 8 and Fig. 9. Note that when the transmission throughput reaches around 250 Mbps, the HAP computational capacity is efficiently utilized with a heavy load of around 90%. Moreover, across all transmission conditions, our heuristic algorithm consistently achieves the best performance. This demonstrates the robustness of our algorithm across varying channel conditions. At the same time, the simplified version of our algorithm, which eliminates the binary search process, achieves the same performance as the full algorithm, while its computation time is the lowest (less than 1 ms). Thanks to the detailed modeling of the queue system and analytical derivation for optimal task rate, our

algorithm can achieve strong performance with an extremely short execution time.

D. Dynamic System Evaluation

To validate our theoretical findings, we construct a realistic simulation environment with real-world parameters that evaluates algorithm performance in terms of HAP computational utilization, i.e., ρ_{comp} and TDC task drop ratio (r_{drop}). We maintain most of our previous configuration in TABLE IV while extending maximum delay tolerance to 0.5s-1.0s to allow greater flexibility. We configure the TDC with $N_{\text{s}}^{\text{TDC}} = 1000$ servers at 30% load. Then, we sample task types with property tuples (b, l, t_{delay}), and generate random tasks evenly for these task types based on TDC load. During the simulation, algorithms will decide whether to offload each task, then we will statistic the performance metric within the simulation system. Each evaluation frame spans 10 seconds and the task is updated every 10 frames.

During each frame, our algorithms analyze task properties for each type and determine the optimal offloading rates (λ_i). The actual offloading decision implements a probabilistic sampling approach: with the calculated optimal offloading rate λ_i and observed total task generation speed λ_i^{total} (obtained through real-time moving average statistics), we establish an offloading probability $r_i^{\text{offload}} = \frac{\lambda_i}{\lambda_i^{\text{total}}}$ for each task type. For comparison, we implement a "Random" baseline that offloads tasks based on the server ratio $r_i^{\text{offload,random}} = \frac{N_{\text{s}}}{N_{\text{s}}^{\text{TDC}}}$.

TABLE V: Comparison of Algorithms: HAP Utilization and Drop Ratio with Static Throughput

Metric	Ours	DE	SQP	Random	RL
ρ_{comp}	90.73%	88.88%	76.55%	6.92%	34.22%
r_{drop}	1.18%	1.18%	1.13%	12.99%	2.16%

TABLE VI: Comparison of Algorithms: HAP Utilization and Drop Ratio with Dynamic Throughput

Metric	Ours	DE	SQP	Random	RL
ρ_{comp}	79.41%	78.15%	59.59%	4.49%	22.84%
r_{drop}	1.25%	1.26%	1.19%	13.25%	2.05%

To compare with the state-of-the-art task offloading method, we develop a Deep Reinforcement Learning (DRL) based offloading agent. The DRL-based method uses a deep neural network to generate offloading decisions (offload or process locally), which is widely used for general sequential decision-making in complex environments [15]–[18]. We applied the Deep Q-learning Network (DQN) based DRL algorithm given our small discrete action space [16].

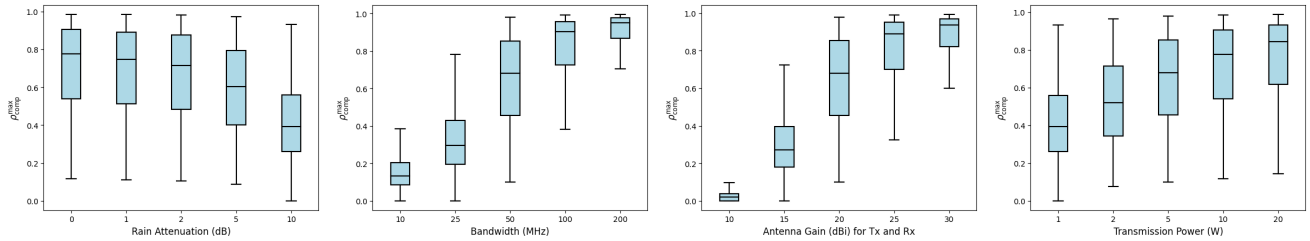
The realistic simulation results demonstrate that our proposed heuristic algorithm achieves strong performance with consistently high HAP utilization (ρ_{comp}) while maintaining very low TDC drop ratios. As shown in Fig. 10 and Fig. 11, the DE algorithm exhibits performance comparable to our heuristic, while SQP demonstrates inconsistent behavior, occasionally yielding invalid results with zero utilization. The random baseline performs the worst, with a significantly higher drop ratio and lower HAP utilization. The RL baseline converges to a state in which the drop ratio is well controlled. However, it fails to achieve high HAP utilization. The RL agent fails to maintain satisfactory system performance under highly dynamic workload variations. The statistical summary in TABLE VI confirms that our algorithm outperforms the alternatives across key metrics.

Although DE achieves competitive performance in terms of utilization and drop ratio, its computational cost renders it impractical for real-time applications, requiring more than 10.0s to compute a solution, exceeding our 10.0s frame duration. In contrast, our heuristic algorithm completes in approximately 1.0s and can achieve sub-millisecond execution times with a simplified version that reduces unnecessary search iterations, as demonstrated in Fig. 5 and Fig. 6. This combination of accuracy and efficiency makes our approach particularly suitable for dynamic edge computing environments with strict timing requirements.

Besides, we also evaluate the robustness of our algorithm when the transmission link between the TDC and the HAP is not stationary. We randomly sample the transmission throughput with a mean value for different time frames, which mimics the throughput fluctuations due to wireless variations. The dynamic of throughput is shown in Fig. 14, and the performance is shown in Fig. 12 and Fig. 13. The performance is similar to that of the ones with static transmission channel conditions. These results confirm that our algorithm maintains robust and efficient performance in such a dynamic environment.

VII. Conclusion

This study investigated task-aware offloading in DC-HAP systems and its potential for reducing energy consumption in terrestrial data centers. Our mathematical analysis demonstrated that transforming the energy minimization objective into computational utilization maximization, using a queueing system model for both transmission and computational processes, provides an effective framework for optimization. The simulation results revealed that task characteristics substantially impact system performance, with the relationship between instruction lengths and bit lengths determining achievable computational utilization in delay-unconstrained scenarios. Furthermore, we proposed three algorithms that optimize hybrid multi-type task offloading, validating the practical value of our analytical framework. In a realistic simulation environment based on real-world parameters, the proposed approach outperformed both traditional and learning-based methods, demonstrating its effectiveness and efficiency in complex dynamic environments. These findings provide valuable insights for the practical implementation of DC-HAP systems and establish a foundation for future research on multi-TDC and multi-HAP architectures, offering a promising direction for next-generation green computing infrastructure.



(a) Sensitivity analysis of rain attenuation on $\rho_{\text{comp}}^{\max}$.

(b) Sensitivity analysis of bandwidth on $\rho_{\text{comp}}^{\max}$.

(c) Sensitivity analysis of antenna gain on $\rho_{\text{comp}}^{\max}$.

(d) Sensitivity analysis of transmit power on $\rho_{\text{comp}}^{\max}$.

Fig. 7: Sensitivity analysis on different link budget components.

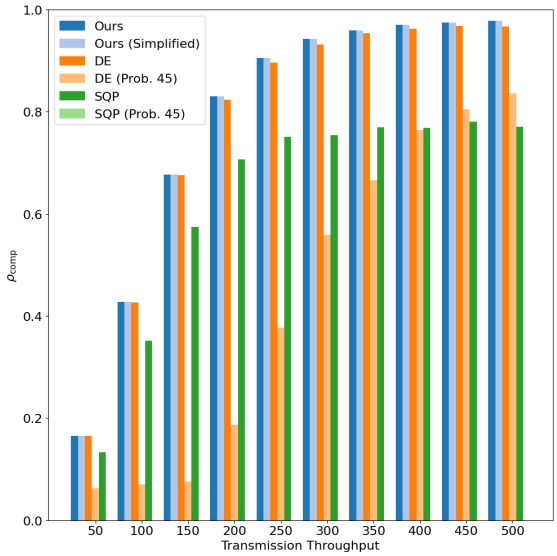


Fig. 8: Sensitivity analysis with various transmission throughputs.

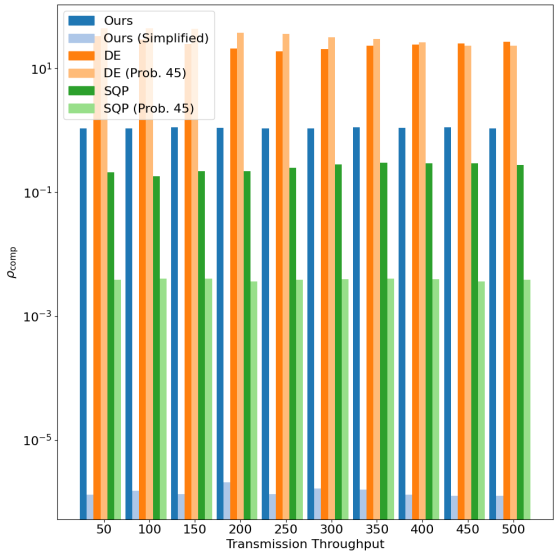


Fig. 9: Execution time with various transmission throughputs.

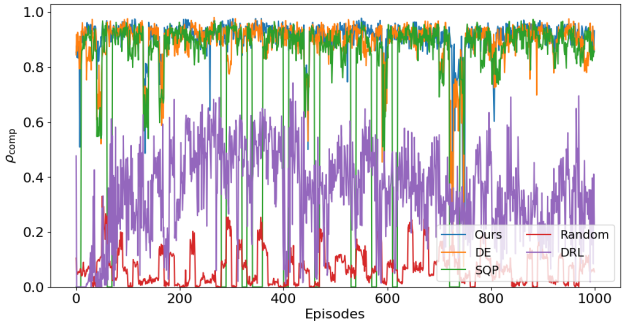


Fig. 10: ρ_{comp} versus time frame under static transmission throughput.

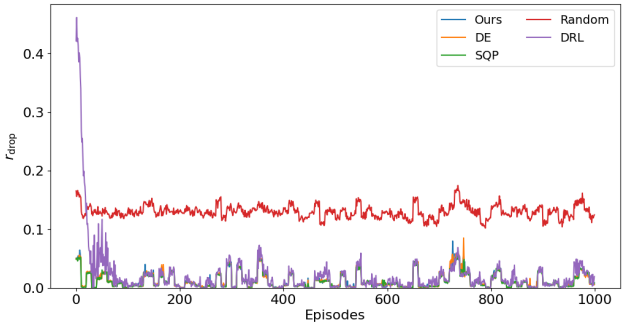


Fig. 11: r_{drop} versus time frame under static transmission throughput.

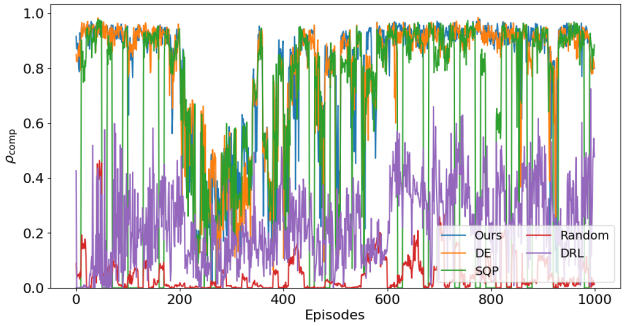


Fig. 12: ρ_{comp} versus time frame under dynamic transmission throughput.

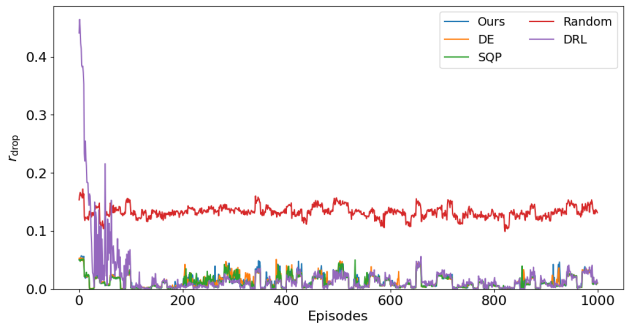


Fig. 13: r_{drop} versus time frame under dynamic transmission throughput.

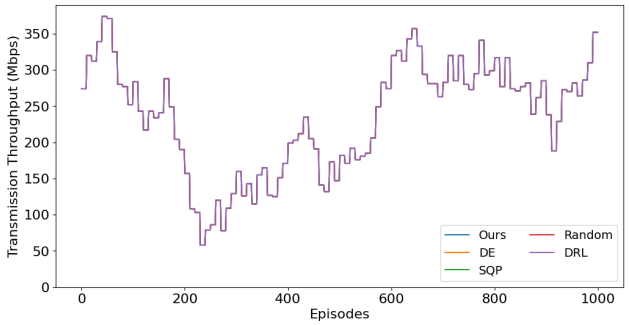


Fig. 14: Transmission throughput versus time.

For practical deployment, several system-level considerations and challenges remain open. First, partitioning TDCs and HAPs based on geographic districts can reduce the complexity of multi-TDC multi-HAP coordination. Second, robust operation requires online adaptation to non-stationary links, weather effects, and intermittent backhaul, together with fault-tolerant mechanisms. Third, deployment cost and scalability should be evaluated. Future research should also extend the current CPU-centric framework to heterogeneous AI workloads and incorporate high-throughput or ultra-reliable links such as optical communication. In addition, learning-based methods with reliability guarantees are needed to improve long-horizon stability and real-time decision quality in large-scale DC-HAP systems.

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, Mar. 2021.

[2] Y. Li, Y. Wen, D. Tao, and K. Guan, "Transforming cooling optimization for green data center via deep reinforcement learning," *IEEE Trans. on cybern.*, vol. 50, no. 5, pp. 2002–2013, Jul. 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, Feb. 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 Commun. and Netw.*, vol. 5, p. 1486488, Sep. 2024.

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

[6] 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 Commun. and Netw.*, vol. 2, p. 716265, Jul. 2021.

[7] W. Abderrahim, O. Amin, and B. Shihada, "Data center-enabled high altitude platforms: A green computing alternative," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 6149–6162, Sep. 2023.

[8] 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 Commun.*, vol. 31, no. 3, pp. 324–331, Jan. 2024.

[9] 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 Trans. on Wireless Commun.*, vol. 21, no. 11, pp. 9010–9024, May 2022.

[10] C. Ding, J.-B. Wang, H. Zhang, M. Lin, and G. Y. Li, "Joint optimization of transmission and computation resources for satellite and high altitude platform assisted edge computing," *IEEE Trans. on Wireless Commun.*, vol. 21, no. 2, pp. 1362–1377, Aug. 2021.

[11] W. Abderrahim, O. Amin, and B. Shihada, "Green quantum computing in the sky," *npj Wireless Technology*, vol. 1, no. 1, p. 5, Nov. 2025.

[12] W. Wu, W. Feng, Y. Fang, Z. Lin, and X. Lu, "Multi-HAP-assisted computation offloading in space-air-ground-sea integrated network," *IEEE Internet of Things J.*, Mar. 2025.

[13] R. Yadav, M. Shafiq, M. Kumar, L. Wei, and M. Abd Elaziz, "Lightweight adaptive learning algorithm for energy-latency tradeoff in IoMT-enabled edge computing," in *17th Int. Conf. Mach. Vision, Edinburgh, UK*, vol. 13517, Oct. 10-13, 2025, pp. 90–97.

[14] W. Li, S. Li, J. Hao, Q. Wu, and R. Wang, "Efficient task offloading and resource allocation in HAPS-assisted LEO satellite networks: A mappo with exact potential game approach," *IEEE Internet of Things J.*, Nov. 2025.

[15] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, "Multi-agent deep reinforcement learning for task offloading in UAV-assisted mobile edge computing," *IEEE Trans. on Wireless Commun.*, vol. 21, no. 9, pp. 6949–6960, Mar. 2022.

[16] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Trans. on Mobile Comput.*, vol. 21, no. 6, pp. 1985–1997, Nov. 2020.

[17] S. A. Zakaryia, M. A. Mead, T. Nabil, and M. K. Hussein, "Task offloading for multi-UAV asset edge computing with deep reinforcement learning," *Cluster Comput.*, vol. 28, no. 7, p. 462, Sep. 2025.

[18] J. Tang, X. Duan, J. Zhou, K. Qu, I. W.-H. Ho, and D. Tian, "Energy-efficient data collection and task offloading optimization in heterogeneous multi-tier UAV systems via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, Oct. 2025.

[19] Q. Fan, W. Zhang, C. Ling, R. Yadav, and H. He, "Data offloading for edge-enabled smart city services via location-functionality correlation analysis," *IEEE Trans. on Services Comput.*, Oct. 2025.

[20] R. Yadav, U. Kumaran, V. Meena, L. Wei, G. Feng, and M. A. Elaziz, "Federated deep learning for malware detection and data protection in edge-enabled IoMT," *Cluster Comput.*, vol. 28, no. 12, p. 757, Nov. 2025.

[21] L. Qin, H. Lu, Y. Chen, B. Chong, and F. Wu, "Towards decentralized task offloading and resource allocation in user-centric MEC," *IEEE Trans. on Mobile Comput.*, May 2024.

[22] H. Vijayaraghavan, J. von Mankowski, and W. Kellerer, "Computifi: Latency-optimized task offloading in multipath multihop LiFi-WiFi networks," *IEEE Open J. of the Commun. Soc.*, Jul. 2024.

[23] W. Li and S. Jin, "Performance evaluation and optimization of a task offloading strategy on the mobile edge computing with edge heterogeneity: W. li, s. jin," *The J. of Supercomputing*, vol. 77, no. 11, pp. 12 486–12 507, Nov. 2021.

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

[25] B. Maruddani and W. Dara, "Ka-band satellite link budget for broadband application in tropical area." *J. Commun.*, vol. 14, no. 7, pp. 622–628, Jul. 2019.

[26] M. S. Islam, J. Sultana, N. P. Lawrence, A. T. Pereira, S. J. Salamon, A. Dinovitser, T. D. Nguyen, J. E. Velazco, B. W.-H. Ng, N. Tansu *et al.*, "Ka-band link budget analysis for deep space communications," *IEEE Access*, Aug. 2025.

[27] I. D. Kristiadi, M. I. Nashiruddin, and M. Sudjai, "High throughput satellite using Ka-band for government multifunctional services in indonesia: Study of link budget and capacity analysis," in *2020 Int. Conf. Radar,*

Antenna, Microwave, Electron. Telecommun., Tangerang, Indonesia, Nov. 18-20, 2020, pp. 85–90.
[28] L. Valentini, A. Faedi, E. Paolini, and M. Chiani, "Analysis of pointing loss effects in deep space optical links," in *2021 IEEE Global Commun. Conf., Madrid, Spain*, Dec. 7-11, 2021, pp. 1–6.
[29] J. R. Barry, E. A. Lee, and D. G. Messerschmitt, *Digital communication*. Springer Science & Business Media, 2012.
[30] R. I. Davis, S. Kollmann, V. Pollex, and F. Slomka, "Controller area network (CAN) schedulability analysis with FIFO queues," in *2011 23rd Euromicro Conf. Real-Time Syst., Porto, Portugal*, Jul. 6-8, 2011, pp. 45–56.
[31] Allen, "Queueing models of computer systems," *Computer*, vol. 13, no. 4, pp. 13–24, Apr. 1980.
[32] J. Anselmi, "Combining size-based load balancing with round-robin for scalable low latency," *IEEE Trans. on Parallel and Distrib. Syst.*, vol. 31, no. 4, pp. 886–896, Oct. 2019.
[33] G. Foschini and J. Salz, "A basic dynamic routing problem and diffusion," *IEEE Trans. on Commun.*, vol. 26, no. 3, pp. 320–327, Jan. 2003.
[34] J. F. Shortle, J. M. Thompson, D. Gross, and C. M. Harris, *Fundamentals of queueing theory*. John Wiley & Sons, 2018.
[35] M. Anagnostou and E. Protonotarios, "Performance analysis of the selective repeat ARQ protocol," *IEEE Trans. on Commun.*, vol. 34, no. 2, pp. 127–135, Jan. 2003.
[36] Z. Lu, W. Wang, and C. Wang, "Modeling, evaluation and detection of jamming attacks in time-critical wireless applications," *IEEE Trans. on Mobile Comput.*, vol. 13, no. 8, pp. 1746–1759, Nov. 2013.
[37] P. J. Burke, "The output of a queuing system," *Operations Res.*, vol. 4, no. 6, pp. 699–704, Dec. 1956.
[38] X. Chao, "On the departure processes of M/M/1/N and GI/G/1/N queues," *Advances in applied probability*, vol. 24, no. 3, pp. 751–754, Sep. 1992.
[39] D. Bertsimas and G. Mourtzinou, "Multiclass queueing systems in heavy traffic: an asymptotic approach based on distributional and conservation laws," *Operations Res.*, vol. 45, no. 3, pp. 470–487, Jun. 1997.
[40] A. Y. Khinchin, D. Andrews, and M. H. Quenouille, *Mathematical methods in the theory of queuing*. Courier Corporation, 2013.
[41] H. Che, Z. Bai, R. Zuo, and H. Li, "A deep reinforcement learning approach to the optimization of data center task scheduling," *Complexity*, vol. 2020, no. 1, p. 3046769, 2020.
[42] A. Marahatta, S. Pirbhulal, F. Zhang, R. M. Parizi, K.-K. R. Choo, and Z. Liu, "Classification-based and energy-efficient dynamic task scheduling scheme for virtualized cloud data center," *IEEE Trans. on Cloud Comput.*, vol. 9, no. 4, pp. 1376–1390, May 2019.
[43] H. Yuan, J. Bi, J. Zhang, and M. Zhou, "Energy consumption and performance optimized task scheduling in distributed data centers," *IEEE Trans. on Syst., man, and Cybern.: Syst.*, vol. 52, no. 9, pp. 5506–5517, Nov. 2021.
[44] N. Padhye, P. Mittal, and K. Deb, "Feasibility preserving constraint-handling strategies for real parameter evolutionary optimization," *Comput. Optim. and Applications*, vol. 62, pp. 851–890, Dec. 2015.
[45] F. Facchinei, L. Lampariello, and G. Scutari, "Feasible methods for non-convex nonsmooth problems with applications in green communications," *Math. Program.*, vol. 164, no. 1, pp. 55–90, Jul. 2017.
[46] F. S. QUADRATIC, "A computationally efficient feasible sequential quadratic programming algorithm," 2000.
[47] O. Exler, T. Lehmann, and K. Schittkowski, "A comparative study of SQP-type algorithms for nonlinear and nonconvex mixed-integer optimization," *Math. Program. Computation*, vol. 4, pp. 383–412, Dec. 2012.
[48] E. Mustafa, J. Shuja, F. Rehman, A. Namoun, M. Bilal, and K. Bilal, "Deep reinforcement learning and SQP-driven task offloading decisions in vehicular edge computing networks," *Comput. Netw.*, p. 111180, May 2025.
[49] H. I. Okagbue, M. O. Adamu, and T. A. Anake, "Differential evolution in wireless communications: A review." *Int. J. of Online & Biomed. Eng.*, vol. 15, no. 11, Nov. 2019.
[50] K. Kashyap, S. Pathak, and N. S. Yadav, "Optimization of spreading code using modified differential evolution for wireless communication," *Wireless Personal Commun.*, vol. 122, no. 2, pp. 1283–1304, Jan. 2022.
[51] H. Guo, J. Liu, and J. Lv, "Toward intelligent task offloading at the edge," *IEEE Netw.*, vol. 34, no. 2, pp. 128–134, Oct. 2019.
[52] U. S. Atmosphere, *US standard atmosphere*. National Oceanic and Atmospheric Administration, 1976.

Jichen Lu received the B.Eng. degree in computer science and technology from Southern University of Science and Technology (SUSTech), China, in 2022, and the M.S. degree in computer science from King Abdullah University of Science and Technology (KAUST), Saudi Arabia, in 2023. He is currently working toward the Ph.D. degree with the Department of Computer, Electrical and Mathematical Science and Engineering, King Abdullah University of Science and Technology (KAUST), Saudi Arabia. His research interests include mobile edge computing, queue theory, machine learning, and reinforcement learning.

Osama Amin received the BSc degree in electrical and electronic engineering from Aswan University, Egypt, in 2000, the MSc degree in electrical and electronic engineering from Assiut University, Egypt, in 2004, and the PhD degree in electrical and computer engineering, University of Waterloo, Canada, in 2010. In 2012, he joined Assiut University as an assistant professor with the Electrical and Electronics Engineering Department. Currently, he is a Senior Research Scientist at King Abdullah University of Science and Technology (KAUST). His research focuses on next-generation communication networks, optical wireless communications, and bio-compatible systems. With a broad background in signal processing and optimization, he aims to improve global connectivity, enhance energy efficiency, and advance healthcare monitoring technologies.

Basem Shihada (M'04-SM'12, IEEE) obtained his Ph.D. in Computer Science from the University of Waterloo, Canada, in 2007. Shortly after completing his studies, Prof. Shihada joined King Abdullah University of Science and Technology as a Founding Faculty member in 2008. His expertise lies in developing cutting-edge wireless systems, where he has made groundbreaking contributions across various domains, including intelligent wireless systems, wireless underwater systems, molecular communication systems, and non-terrestrial systems. Prof. Shihada's notable achievements are the creation and successful demonstration of Aqua-Fi, the world's first underwater Wi-Fi. Sun-Fi, the world first communication via building glass, and communication via breath. Prof. Shihada's work has been recognized with several best paper awards at renowned conferences within his field. His invaluable contributions have also been published in prestigious scientific journals such as Nature Electronics and many IEEE Transactions. In 2023, he become an area editor and received the exemplary editor award from the IEEE Communications Letter journal.