Distributed Computational Offloading Across Multiple HAPs and Terrestrial Data Centers

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

***Abstract*—Data Center-enabled High Altitude Platforms (DC-HAPs) show great potential in reducing the energy consumption of terrestrial data centers by leveraging natural cooling and solar power harvesting. This paper advances the field by analyzing multi-platform architectures with multiple terrestrial data centers (TDCs) and HAPs performing heterogeneous task offloading. We formulate this complex problem mathematically and develop a feasibility-preserving transformation approach to ensure optimization stability. To solve this challenging problem, we propose a novel heuristic sequential algorithm specifically designed for multi-platform DC-HAP systems, while also tailoring two established optimization methods—sequential quadratic programming (SQP) and differential evolution (DE)—to address our unique problem constraints. Comprehensive simulations demonstrate that robust results are guaranteed with our proposed heuristic algorithm, while the SQP algorithm with the transformed problem balances performance and runtime better than the DE approach. The increasing number of HAPs leads to capability degradation for the SQP and DE algorithms, while our proposed heuristic algorithm maintains powerful performance with the highest efficiency for complex system architectures.**

***Index Terms*—multiple high altitude platforms (HAPs), data center-enabled HAP, green computing network, non-terrestrial network, task-aware offloading, queue delay**

I. Introduction

The exponential growth in computational demands from artificial intelligence, edge computing, and data analytics has created an urgent need for energy-efficient computing infrastructure. Data center electricity consumption, which represented 1.3% of global usage in 2010, is projected to reach 10% by 2050 [1]. This trajectory raises critical concerns about both resource availability and environmental impact in the future communication and computing system [2].

Cooling systems constitute 30-50% of data center energy consumption due to the need for thermal management of densely packed computing resources, while computational processing itself typically accounts for 20-30% [3], [4]. Data Center-enabled High Altitude Platforms (DC-HAPs) address this challenge through strategic positioning in the stratosphere, where ambient temperatures naturally eliminate cooling requirements while abundant solar exposure enables renewable energy harvesting. This dual advantage creates opportunities for significantly more sustainable computing infrastructures.

Research on aerial computing platforms has evolved across several dimensions, and plays an important role in the next generation communication system by providing multiple services from an aerial level [5]–[7]. Ren *et al.* explored the caching and computation assistance for transportation systems with HAPs to achieve lower user delay [8]. Ding *et al.* investigated task processing optimization in hybrid HAP-satellite architectures, though limited to single-task scenarios [9]. Abderrahim *et al.* examined DC-HAP energy efficiency, establishing their potential but focusing exclusively on single-platform deployments [4]. We have investigated a comprehensive analysis of single DC-HAP systems with single and multi-type workloads, developing analytical expressions for maximum computational utilization and establishing the relationship between task characteristics and system performance [10].

However, practical implementations will inevitably involve multiple platforms serving diverse geographic regions and workload requirements. This paper extends our analysis to the more general and realistic scenario of multiple interconnected TDC-HAP systems with heterogeneous task offloading requirements. Our work provides a foundation for the practical deployment of aerial computing infrastructures at scale, addressing both technical optimization challenges and system design considerations. Our contributions include:

- Extending our theoretical framework to accommodate multiple HAPs and TDCs with diverse communication pathways and computational capabilities.
- Formulating a transformation-based optimization framework that guarantees solution feasibility despite the interconnected constraints of multi-platform architectures.
- Developing algorithms tailored to the unique topology and resource allocation challenges of distributed aerial computing networks.

II. System Modeling

In our multi-platform architecture, TDCs can offload computational tasks to HAPs equipped with data center capabilities. We focus on uplink transmission and aerial computational processing, since the result data generated by DC-HAPs is generally compact and requires minimal resources for downlink delivery to TDCs [11], [12].

A. Computational Energy Model

Consider a heterogeneous system comprising N_{TDC} TDCs and N_{HAP} HAPs, both capable of processing computational workloads. Each HAP h contains N_{s} identical servers, each providing computational capacity of μ instructions per second (IPS). Each TDC t processes N^t different task types, with all tasks initially arriving at TDCs before potential HAP offloading.

For each task type $i \in 1, \ldots, N^t$ at TDC t, we define its instruction length l_i^t (instructions/task) and bit length b_i^t (bits/task). The offloading rate of type i tasks from TDC t to HAP h is denoted by $\lambda_i^{t,h}$. According to [10], the energy consumption in TDCs is proportional to the utilization of TDC servers. Therefore, we convert the energy saving metric to computational utilization of HAPs and maximize it:

$$\rho_{\mathrm{comp}}^{\mathrm{total}} = \sum\nolimits_{h=1}^{N_{\mathrm{HAP}}} \frac{\sum_{t=1}^{N_{\mathrm{TDC}}} \sum_{i=1}^{N^t} \lambda_i^{t,h} l_i^t}{N_{\mathrm{s}}\mu}. \tag{1}$$

B. Transmission Model

For the wireless communication between TDCs and HAPs, we adopt the transmission throughput model from [10]. The conservative data rate of link between TDC t and HAP h is:

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

where B is the bandwidth; N_{r} is the number of transmitter antennas in MIMO; and γ is the path loss.

C. Tasks Offloading Delay

The multi-platform scenario introduces additional complexity since each TDC can offload any task type to any HAP within the network. Building upon [10], we extend the delay expressions to accommodate this topology. For the i-th task type from TDC t offloaded to HAP h, the total sojourn time is:

$$T_i^{t,h} = T_{\mathrm{trans},i}^{t,h} + T_{\mathrm{comp},i}^{t,h}, \tag{3}$$

where each component incorporates both waiting time W and service time S:

$$T_{\mathrm{trans},i}^{t,h} = W_{\mathrm{trans},i}^{t} + S_{\mathrm{trans},i}^{t,h} \tag{4}$$

$$T_{\mathrm{comp},i}^{t,h} = W_{\mathrm{comp},i}^{h} + S_{\mathrm{comp},i}^{t,h}. \tag{5}$$

Note that the waiting time depends on the specific transmission or computational queue. For transmission, we derive:

$$W_{\mathrm{trans},i}^{t} \approx \frac{\sum_{h=1}^{N_{\mathrm{HAP}}} \sum_{j=1}^{N^t} \lambda_j^{t,h} \left(1 + c_{\mathrm{s,trans},j}^2\right) \left(\frac{b_j}{R}\right)^2}{2(1-\rho_{\mathrm{trans}})}, \tag{6}$$

$$S_{\mathrm{trans},i}^{t,h} = \frac{b_i^t}{R^{t,h}}. \tag{7}$$

The transmission resource utilization must satisfy condition:

$$\rho_{\mathrm{trans}}^{t} = \sum\nolimits_{h=1}^{N_{\mathrm{HAP}}} \sum\nolimits_{i=1}^{N^t} \frac{\lambda_i^{t,h} b_i^{t,h}}{R^{t,h}} < 1. \tag{8}$$

Similarly for computation, with $\rho_{\mathrm{comp}} = \sum_{j=1}^{N} \rho_{\mathrm{comp},j}$ and $c_{\mathrm{s,comp},j}$ as the coefficient of variation of computational service time for type j:

$$W_{\mathrm{comp},i}^{h} \approx \frac{\sum_{t=1}^{N_{\mathrm{TDC}}} \sum_{j=1}^{N^t} \lambda_j^{t,h} \left(1 + c_{\mathrm{s,comp},j}^2\right) \left(\frac{l_j}{N_{\mathrm{s}}\mu}\right)^2}{2(1-\rho_{\mathrm{comp}})}, \tag{9}$$

$$S_{\mathrm{comp},i}^{t,h} = \frac{l_i^t}{\mu}. \tag{10}$$

The computational resource utilization must satisfy:

$$\rho_{\mathrm{comp}}^{h} = \sum\nolimits_{t=1}^{N_{\mathrm{TDC}}} \sum\nolimits_{i=1}^{N^t} \frac{\lambda_i^{t,h} l_i^t}{N_{\mathrm{s}}\mu} < 1. \tag{11}$$

III. Optimization Problem Formulation

In this section, we extend our analysis to a more practical scenario with multiple TDCs and HAPs. We consider a fully connected network architecture where each TDC can offload tasks to any HAP, and each HAP can receive and process tasks from any TDC. We first formulate the general optimization problem and then apply a feasibility-preserving transformation to ensure solution stability.

A. Problem Formulation

For our multiple TDC-HAP scenario, we assume known positions of all TDCs and HAPs. Our objective is to determine the optimal offloading rates $\lambda_i^{t,h}$ across the network that maximize computational utilization. Task properties (b_i^t, l_i^t) are predetermined by application requirements [13]–[15].

Following the modeling, we formulate the optimization problem to maximize computational resource utilization while ensuring system stability and meeting latency constraints:

$$\max_{\lambda_i^{t,h}} \quad \rho_{\mathrm{comp}}^{\mathrm{total}} \left(\lambda_1^{1,1}, \ldots, \lambda_{N^{N_{\mathrm{TDC}}}}^{N_{\mathrm{TDC}},N_{\mathrm{HAP}}}\right) \tag{12a}$$

$$\text{subject to} \quad \lambda_i^{t,h} \geq 0, \, \forall t, h, i, \tag{12b}$$

$$\rho_{\mathrm{trans}}^{t} \left(\lambda_1^{t,1}, \ldots, \lambda_{N^t}^{t,N_{\mathrm{HAP}}}\right) < 1, \, \forall t, \tag{12c}$$

$$\rho_{\mathrm{comp}}^{h} \left(\lambda_1^{1,h}, \ldots, \lambda_{N^{N_{\mathrm{TDC}}}}^{N_{\mathrm{TDC}},h}\right) < 1, \, \forall h, \tag{12d}$$

$$T_i^{t,h} \left(\lambda_1^{1,1}, \ldots, \lambda_{N^{N_{\mathrm{TDC}}}}^{N_{\mathrm{TDC}},N_{\mathrm{HAP}}}\right) \leq t_{\mathrm{delay}}, \, \forall t, h, i. \tag{12e}$$

where constraints (12c) and (12d) ensure stable transmission and computation queues for every TDC and HAP, respectively, while (12e) enforces latency requirements for all task types across all network paths.

B. Feasibility-preserving Transformation

The multi-platform formulation in (12) presents greater complexity and higher variable correlation than our single TDC-HAP model [10]. This increased interconnectedness raises the likelihood of encountering infeasible intermediate solutions during optimization. We therefore adapt our feasibility-preserving transformation approach to deal with this complicated scenario.

Our transformation follows three steps. First, we convert variables $\lambda_i^{t,h}$ to utilization pairs $(u_i^{t,h}, v_i^{t,h})$, where $u_i^{t,h}$ represents transmission utilization and $v_i^{t,h}$ denotes computational utilization for each task type. Second, we eliminate dependency between these variables by defining:

$$\lambda_i^{t,h} = \min\left(\frac{R^{t,h}u_i^{t,h}}{b_i^{t,h}}, \frac{\mu N_{\mathrm{s}} v_i^{t,h}}{l_i^{t,h}}\right). \tag{13}$$

Finally, we apply a Softmax-based feasibility-preserving transformation. Unlike the single TDC-HAP case, we must account for separate utilization constraints across platforms. From (8) and (11), transmission utilization is bounded per TDC while computational utilization is bounded per HAP. We introduce slack variables for each TDC and HAP, transforming unconstrained variables $(u_i'^{t,h}, v_i'^{t,h})$ into feasible utilization pairs:

$$u_i^{t,h} = \mathcal{S}^{\mathrm{t}}(\mathbf{u}') = \frac{e^{u_i'^{t,h}}}{\sum_{h=1}^{N_{\mathrm{HAP}}}\sum_{j=0}^{N^t} e^{u_j'^{t,h}} + e^{u_0'^{t}}} \tag{14a}$$

$$v_i^{t,h} = \mathcal{S}^{\mathrm{c}}(\mathbf{v}') = \frac{e^{v_i'^{t,h}}}{\sum_{t=1}^{N_{\mathrm{TDC}}}\sum_{j=0}^{N^t} e^{v_j'^{t,h}} + e^{v_0'^{h}}} \tag{14b}$$

where $e^{u_0'^{t}}$ and $e^{v_0'^{h}}$ serve as slack variables for each TDC t and HAP h, respectively.

After the transformation, we can write the problem as:

$$\max_{u_i'^{t,h}, v_i'^{t,h}} \rho_{\mathrm{comp}}^{\mathrm{total}} \tag{15a}$$

$$\text{s.t. } T_i^{t,h} \le t_{\mathrm{delay}} \tag{15b}$$

After solving the problem, we can we can calculate λ_i from:

$$\lambda_i^{t,h} = \min\left(\frac{R}{b_i}\frac{e^{u_i'^{t,h}}}{\sum_{h=1}^{N_{\mathrm{HAP}}}\sum_{j=0}^{N^t} e^{u_j'^{t,h}} + e^{u_0'^{t}}}, \frac{\mu N_s}{l_i}\frac{e^{v_i'^{t,h}}}{\sum_{t=1}^{N_{\mathrm{TDC}}}\sum_{j=0}^{N^t} e^{v_j'^{t,h}} + e^{v_0'^{h}}}\right). \tag{16}$$

IV. Algorithms Design

This section introduces three algorithms to solve Problem 15: a heuristic approach, a gradient-based method (SQP), and an evolutionary algorithm (DE).

A. Heuristic Sequential Algorithm

Our heuristic approach first orders task types by their maximum achievable computational utilization before sequentially allocating resources. However, in multi-platform scenarios, the variables $\lambda_i^{t,h}$ are highly correlated through utilization constraints, complicating direct extension of this approach. We therefore develop a link selection strategy that decomposes our problem into several single TDC-HAP subproblems.

Our heuristic algorithm selectively activates certain links while disabling others, despite full connectivity being physically available. The selection process follows three principles: (1) prioritize links with higher throughput to maximize transmission capacity, (2) avoid assigning multiple HAPs to a single TDC to prevent transmission bottlenecks, and (3) balance throughput allocation across HAPs by prioritizing those with lower aggregate throughput. Algorithm 1 details this approach, which selects $\min(N_{\mathrm{TDC}}, N_{\mathrm{HAP}})$ links in total.

Algorithm 1 Heuristic Sequential Algorithm

Initialization: Create an empty selected HAP index vector $\mathbf{S}$ with length of N_{TDC}, initialize all the elements as -1.
Begin:
  Sum the transmission throughput of each HAP, and sort the HAPs by the sum of the throughput from low to high.
  for h in N_{HAP} **do**
    Select the non-assigned TDC with the highest throughput to the HAP h, and assign the HAP index to the TDC.
  end for
  Get the selected HAP index vector $\mathbf{S}$.
  for t, h in $enumurate(\mathbf{S})$ **do**
    Create an empty selected type list S^t. Create an empty task rate vector λ^t. Initialize $\rho_{\mathrm{comp}}^h = 0$.
    Calculate the $\lambda_{\max,i}^{t,h}, \rho_{\mathrm{comp},i}^{t,h}$ using derived equations in [10]. Sort types based on $\rho_{\mathrm{comp},i}^{t,h}$, put to list L.
    for i in N^t **do**
      if $i == 1$ **then**
        Append i to S^t; append $\lambda_{\max,i}^{t,h}$ to λ^t; $\rho_{\mathrm{comp}}^h = \rho_{\mathrm{comp},i}^{t,h}$.
      else
        for j in n_{step} **do**
          Set $r_{\mathrm{s}} = \frac{j}{n_{\mathrm{step}}}$; $\lambda_{\mathrm{prev}}^t = r_{\mathrm{s}}\lambda^t$.
          Calculate $\lambda_{i,r_{\mathrm{s}}}^{t,h}$ as in [10].
          Re-scale as $r_{\mathrm{b}}[\lambda_{\mathrm{prev}}^t, \lambda_{i,r_{\mathrm{s}}}^t]$ with binary search to guarantee (12e), and calculate ρ'_{comp} with it.
          if $\rho'_{\mathrm{comp}} \ge \rho_{\mathrm{comp}}^h$ **then**
            $\lambda_{\mathrm{new}}^t = r_{\mathrm{b}}[\lambda_{\mathrm{prev}}^t, \lambda_{i,r_{\mathrm{s}}}^t]$; $\rho_{\mathrm{comp}}^{\max} = \rho'_{\mathrm{comp}}$.
          end if
        end for
        Set $\lambda^t = {\lambda_{\mathrm{new}}}^t$; append type i into the list S^t.
      end if
    end for
    Get the λ^t using (16) and calculate ρ_{comp}^h as in [10].
  end for
End: Calculate $\rho_{\mathrm{comp}}^{\mathrm{total}}$ with (1).

Complexity Analysis: The original single TDC-HAP heuristic algorithm has complexity $O(N^t n_{\mathrm{step}} \log n_{\mathrm{step}})$, where N^t is the number of task types and n_{step} is the linear search step count [10]. Our multi-platform extension adds link selection complexity of $O(N_{\mathrm{HAP}}N_{\mathrm{TDC}} + N_{\mathrm{HAP}}N_{\mathrm{TDC}}\log N_{\mathrm{TDC}} + N_{\mathrm{HAP}}\log N_{\mathrm{HAP}})$, where terms correspond to throughput calculation, per-HAP sorting, and HAP aggregate throughput sorting, respectively. The overall complexity is $O(N_{\mathrm{HAP}}N_{\mathrm{TDC}}\log N_{\mathrm{TDC}} + N_{\mathrm{HAP}}\log N_{\mathrm{HAP}} + \min(N_{\mathrm{TDC}}, N_{\mathrm{HAP}})\max(N^t) n_{\mathrm{step}}\log n_{\mathrm{step}})$.

B. Sequential Quadratic Programming Algorithm

We implement Sequential Quadratic Programming (SQP) to solve our non-linear, non-convex optimization problem. We first transform the original problem as shown in (15) to ensure utilization constraints are inherently satisfied.

For each iteration, we construct the Lagrangian function, solve the quadratic programming subproblem, and update variables until convergence or reaching the maximum iteration count m_{iter}. The multi-platform scenario expands variables from a vector of length N^t to a tensor of size $N_{\text{TDC}} \times N_{\text{HAP}} \times N^t$, making our feasibility-preserving transformation particularly important.

Complexity Analysis: The complexity of the original SQP algorithm is $O(m_{\text{iter}} {N^t}^3)$, where m_{iter} is the maximum iteration limit, and N^t is the number of types of tasks in the set [10]. Since the only difference between the single TDC-HAP single-type scenario and the multiple TDC-HAP multi-type scenario is the dimension of the variables and constraints, the complexity of the algorithms will change the original dimension N^t to $N' = N_{\text{TDC}} \times N_{\text{HAP}} \times \max(N^t)$. Therefore, the complexity of the algorithms will be $O(m_{\text{iter}} {N'}^3)$ for the SQP algorithm for multiple TDC-HAP multi-type scenario.

Algorithm 2 Sequential Quadratic Programming

1: **Initialization:** u'_i, v'_i: initial guess for transmission and computation utilization of each type i; k: initial guess for the Lagrange multiplier; $c_{\text{iter}} = 0$: iteration counter.
2: **Begin:**
3: **while** $c_{\text{iter}} < m_{\text{iter}}$ **do**
4: Transform ${u'}_i^{t,h}$, ${v'}_i^{t,h}$ to $u_i^{t,h}$, $v_i^{t,h}$ with Eq. (14). Get the $\lambda_i^{t,h}$ from (16) and identify the minimum expression which will be used to calculate the gradient.
5: Construct Lagrangian function as:

$$L = \rho_{\text{comp}}^{\text{total}} + k(\mathbf{T} - t_{\text{delay}}) \tag{17}$$

6: Calculate the gradient g of the Lagrangian function with respect to $\mathbf{u}$ and $\mathbf{v}$; construct the Hessian matrix H of the Lagrangian function.
7: Construct and solve the QP subproblem to obtain the search direction p:

$$\begin{aligned} &\min_p \ \frac{1}{2} p^T H p + g^T p \\ &\text{s.t. } (\mathbf{T} - t_{\text{delay}}) + \nabla(\mathbf{T} - t_{\text{delay}})^T p \le 0 \end{aligned} \tag{18}$$

8: Perform a line search to determine the step size α.
9: Update the current point:

$${u'}_i^{t,h} = {u'}_i^{t,h} + \alpha p, \ {v'}_i^{t,h} = {v'}_i^{t,h} + \alpha p \tag{19}$$

10: Update the Lagrange multiplier k. $c_{\text{iter}} = c_{\text{iter}} + 1$.
11: Break if $||p|| < \text{tol}$.
12: **end while**
13: **End:** Get the vector λ with (16), calculate $\rho_{\text{comp}}^{\text{total}}$ with (1).

C. Differential Evolution Algorithm

For our non-linear, non-convex optimization problem, we also implement Differential Evolution (DE), which excels where traditional gradient-based methods may struggle. DE's population-based approach explores multiple solution space regions simultaneously, making it less sensitive to initial conditions and well-suited for complex landscapes.

We first transform the problem to ensure constraint satisfaction during initial population sampling. The DE algorithm then iteratively refines candidate solutions through selection, mutation, and crossover operations, evaluating fitness based on objective function value and constraint satisfaction.

Complexity Analysis: The complexity of the DE algorithm for single TDC-HAP multi-type is $O(m_{\text{iter}}(N_{\text{pop}} N^t + N_{\text{pop}}^2))$, 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 [10]. The multiple TDC-HAP scheme changes the original dimension from N^t to $N' = N_{\text{TDC}} \times N_{\text{HAP}} \times N$. Therefore, the complexity of the algorithms will be $O(m_{\text{iter}}(N_{\text{pop}} N' + N_{\text{pop}}^2))$ for the Differential Evolution algorithm.

Algorithm 3 Differential Evolution Algorithm

1: **Initialization:** $({u'}_i^{t,h}, {v'}_i^{t,h})$: initial guess of N_{pop} individuals for transmission utilization and computation utilization of each type i; $c_{\text{iter}} = 0$: iteration counter.
2: **Begin:**
3: **while** $c_{\text{iter}} < m_{\text{iter}}$ **do**
4: Transform ${u'}_i^{t,h}$, ${v'}_i^{t,h}$ to $u_i^{t,h}$, $v_i^{t,h}$ with (14) for each individual. Get the $\lambda_i^{t,h}$ with (16).
5: Evaluate the objective function $\rho_{\text{comp}}^{\text{total}}$ for each individual; evaluate the constraint (15b) for each individual.
6: Select the best individual $(\mathbf{u}_{\text{best}}, \mathbf{v}_{\text{best}})$ from the population based on the following rules: select the feasible ones with higher objective value; if no feasible ones, select ones with lower constraint violation value.
7: Select two random individuals $(\mathbf{u}_b, \mathbf{v}_b)$ and $(\mathbf{u}_c, \mathbf{v}_c)$ from the population.
8: Generate a trial vector as:

$$\mathbf{u}_t = \mathbf{u}_{\text{best}} + F(\mathbf{u}_b - \mathbf{u}_c) \tag{20a}$$

$$\mathbf{v}_t = \mathbf{v}_{\text{best}} + F(\mathbf{v}_b - \mathbf{v}_c) \tag{20b}$$

9: Perform a crossover operation as to generate new individual $(\mathbf{u}_{\text{new}}, \mathbf{v}_{\text{new}})$ for each individual.
10: Evaluate the objective function $\rho_{\text{comp}}^{\text{total}}$ and the constraint Eq. (15b) for the new individual $(\mathbf{u}_{\text{new}}, \mathbf{v}_{\text{new}})$; replace the old individual $(\mathbf{u}, \mathbf{v})$ with the new individual $(\mathbf{u}_{\text{new}}, \mathbf{v}_{\text{new}})$ if the new individual has a better performance based on the rules above.
11: $c_{\text{iter}} = c_{\text{iter}} + 1$.
12: **end while**
13: **End:** Get the vector λ with (16), calculate $\rho_{\text{comp}}^{\text{total}}$ with (1).

V. Simulation Results

Our simulation considers multiple TDCs and HAPs positioned within a region of 1 degree in latitude and 1 degree in longitude. In each single experiment, we generate the positions of TDCs and HAPs randomly within the region to evaluate the performance of algorithms under arbitrary topologies. The transmission configuration includes $P_{\text{trans}} = 100$W, $N_{\text{t}} = 2$ transmitting antennas, a carrier frequency of $f_{\text{c}} = 31$GHz, and bandwidth $B = 100$MHz. Each HAP contains $N_{\text{s}} = 20$ servers processing at $\mu = 580$MIPS [16], with tasks evenly distributed. TDCs are equipped with $N_{\text{r}} = 16$ receiving antennas. Task data sizes range from 300KB to 800KB, with computational requirements between 100 and 1000 MegaCycles and an average cycles-to-instructions ratio of 3.7 [17]. Note that the summation $\rho_{\text{comp}}^{\text{total}}$ could be greater than 1.

A. Simulation Performance with Varying Type Number

We first evaluate algorithm performance with different numbers of task types. The setup includes 2 TDCs and 2 HAPs, with task types varying from 1 to 5 per TDC (each TDC having 5 potential task types). For fair comparison, we order types using Algorithm 1 across all algorithms. Results appear in Fig. 1.

Our heuristic algorithm shows remarkable stability, maintaining consistent performance as complexity increases. SQP and DE algorithms achieve higher utilization with fewer task types, suggesting the heuristic finds local optima. However, both SQP and DE exhibit performance degradation as task types increase, indicating sensitivity to problem dimensionality. Notably, there exists a rise for the SQP and DE algorithms when the type number is 5. Since we add the types in the same order as in the heuristic algorithm for comparison fairness, this suggests that the types with low utilization order may benefit the final performance when combined with other high-order types.

The transformed Problem (15) generally improves SQP and DE performance by reducing infeasible intermediate solutions. When the original Problem (12) performs better with fewer task types, this likely stems from its lower dimensionality facilitating convergence. As dimensions increase, the original problem becomes harder to solve, particularly for the SQP algorithm, which frequently encounters infeasible solutions and fails to converge without transformation.

As for the runtime of the algorithms, we show the time plot in Fig. 2. The SQP algorithm is the fastest due to its gradient-based design, which converges rapidly. The heuristic algorithm is the second fastest, thanks to its simple searching design. However, note that the performance of the heuristic algorithm does not drop with fewer task types; we can just apply the results of it with one task type to save time. The DE algorithm is the slowest with population-based design. Combining Fig. 1, SQP is recommended to be combined with type sorting in Algorithm 1.

B. Simulation Performance with Varying Number of HAPs

We also explore how the number of HAPs affects the performance of our algorithms. The simulation setup includes

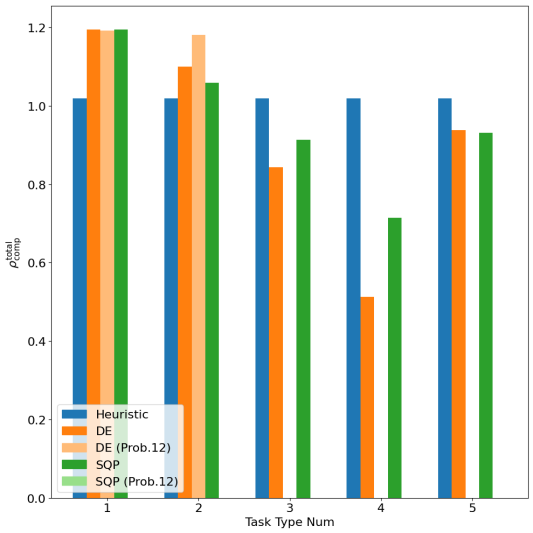

Fig. 1: $\rho_{\text{comp}}^{\text{total}}$ with various numbers of task types.

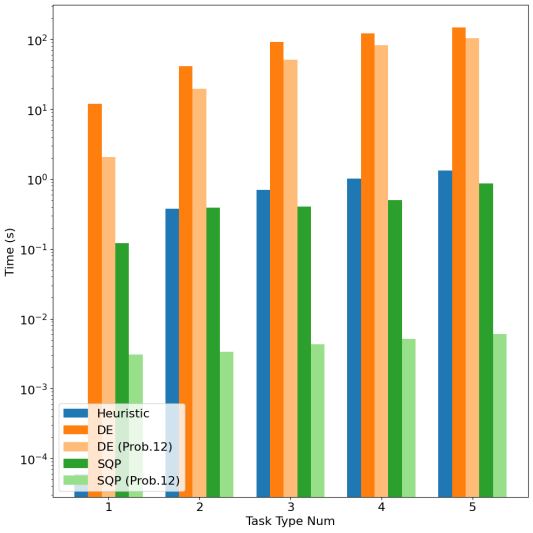

Fig. 2: Algorithm runtime with various numbers of task types.

2 TDCs, and the number of HAPs is varied from 2 to 8, where the new HAPs' positions are randomly generated as before. The number of task types is fixed at 2.

From Fig. 3, we can see that the total utilization increases with the number of HAPs. This is because more HAPs provide more computational resources, and the transmission resources are also increased due to the increased number of links and the possibility of gaining higher throughput. Our heuristic algorithm maintains the increasing trend, which suggests that our heuristic design is robust when dealing with the complex topology scheme of the DC-HAP system. The SQP algorithm failed to achieve high total computational utilization when the number of HAPs is large due to the difficulty of finding a good local optimum under rapid growth of complexity. At the same time, the DE algorithm maintains an increasing performance comparable to our heuristic algorithm and surpasses our

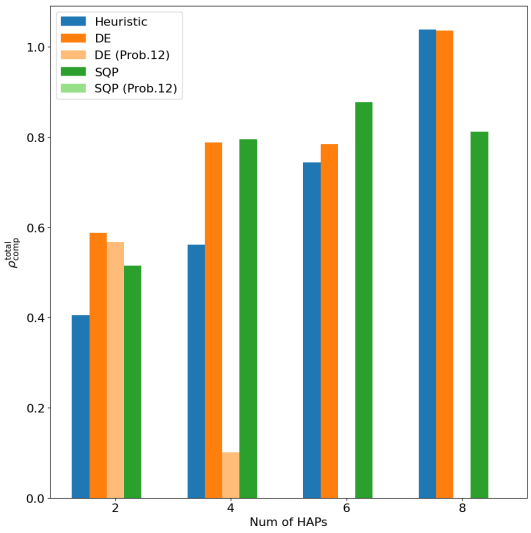

Fig. 3: $\rho_{\text{comp}}^{\text{total}}$ with various numbers of HAPs.

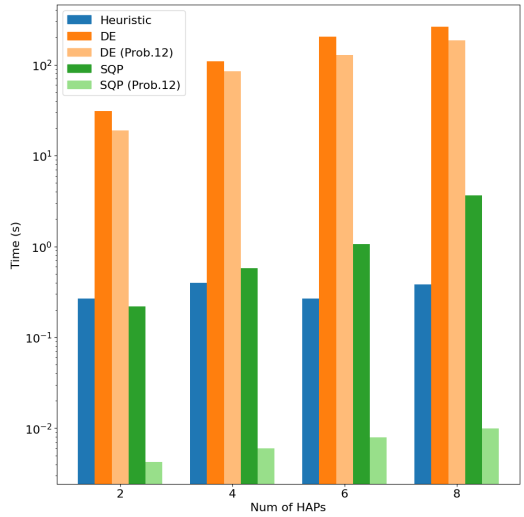

Fig. 4: Algorithm runtime with various numbers of HAPs.

algorithm when the number of HAPs is small. This indicates the broader searching ability of the DE algorithm for multiple TDC-HAP multi-type scenarios, while being sensitive to high-dimensional complexity. Combining with the runtime shown in Fig. 4, our heuristic algorithm is the most efficient algorithm with powerful performance under complex system topologies.

VI. Conclusion

This work extends task-aware offloading in DC-HAP systems to multi-platform scenarios by developing a mathematical framework that transforms complex optimization problems into more tractable forms. We introduced three algorithms—a stable heuristic approach, a fast gradient-based SQP method, and a high-performance evolutionary DE algorithm—each offering different trade-offs between computational efficiency and solution quality. Our simulations reveal that while the heuristic maintains consistent performance across varying conditions, DE and SQP can achieve higher utilization in simpler scenarios but show greater variability with increased complexity. We observed decreasing capability for the SQP and DE algorithms with additional HAPs, while our heuristic algorithm maintains a robust and efficient performance. These findings provide valuable insights for designing next-generation green computing infrastructure, balancing performance demands with resource constraints while maintaining service guarantees in distributed aerial computing systems.

References

[1] 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.

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

[3] L. Ismail and H. Materwala, "Computing server power modeling in a data center: Survey, taxonomy, and performance evaluation," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–34, June 2020.

[4] 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, July 2023.

[5] 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, July 2021.

[6] 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.

[7] S. Ammar, C. P. Lau, and B. Shihada, "An in-depth survey on virtualization technologies in 6g integrated terrestrial and non-terrestrial networks," *IEEE Open J. Commun. Soc*, vol. 5, pp. 3690–3734, June 2024.

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

[9] 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. Wireless Commun.*, vol. 21, no. 2, pp. 1362–1377, Aug. 2021.

[10] 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

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

[12] H. Vijayaraghavan, J. von Mankowski, and W. Kellerer, "ComputiFi: Latency-optimized task offloading in multipath multihop LiFi-WiFi networks," *IEEE Open J. Commun. Soc*, July 2024.

[13] 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.

[14] 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. Cloud Comput.*, vol. 9, no. 4, pp. 1376–1390, May 2019.

[15] H. Yuan, J. Bi, J. Zhang, and M. Zhou, "Energy consumption and performance optimized task scheduling in distributed data centers," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 52, no. 9, pp. 5506–5517, Nov 2021.

[16] W. Abderrahim, O. Amin, and B. Shihada, "Data center-enabled high altitude platforms: A green computing alternative," *IEEE Trans. Mobile Comput.*, Sep. 2023.

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