This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TVT.2025.3612207, IEEE Transactions on Vehicular Technology

LLM-based Task Offloading and Resource Allocation in Satellite Edge Computing Networks

Minghao Sun, Jinbo Hou, Kehai Qiu, *Member, IEEE,* Kezhi Wang, *Senior Member, IEEE,* Xiaoli Chu, *Senior Member, IEEE,* Zitian Zhang

***Abstract*—Satellite Mobile Edge Computing (MEC) networks offer a promising solution for delivering global services to terrestrial Internet of Things (IoT) terminals in 5G and beyond. However, satellite MEC systems face challenges such as under-utilization of resources and task congestion, leading to resource waste and increased latency. In this paper, we investigate the joint resource allocation and task offloading problem in multi-satellite MEC networks, aiming to minimize the average latency of IoT terminals. To solve the joint optimization problem involving IoT terminals' task offloading decisions, uplink transmission power and sub-channel allocation, and satellite computation resource allocation, we propose an iterative optimization algorithm that uses the Lagrange multipliers method to optimize the satellite computation resource allocation and a Large Language Model (LLM) based optimizer to optimize the other variables in each iteration. Prompts and templated parameters are designed to enhance the LLM's inference accuracy and generalization capability across scenarios with varying numbers of satellites and IoT terminals. Simulation results show that our proposed LLM-based algorithm outperforms benchmark algorithms in convergence speed and average latency of IoT terminals.**

***Index Terms*—Satellite mobile edge computing, task offloading, resource allocation, Large Language Model, Internet of Things.**

I. Introduction

INTERNET of Things (IoT) terminals have driven numerous intelligent applications [1]. However, terrestrial communication networks fail to provide reliable communication services for IoT terminals in remote areas, such as disaster zones, oceans, and deserts. Low-Earth-Orbit (LEO) satellite Mobile Edge Computing (MEC) networks can help provide global service coverage for IoT terminals [2], [3]. Nonetheless, achieving efficient resource allocation and task offloading while meeting low-latency requirements remains challenging due to limited communication and computation resources at both satellites and terminals [4].

Some research has been conducted in this area. The authors in [5] minimized latency and energy consumption in a satellite MEC network by Breadth-First Search and greedy algorithms. The latency was minimized by using the Genetic Algorithm (GA) and Lagrange multiplier method in [6] and by employing game theory and many-to-one matching theory in [7]. The authors in [8] solved a weighted-sum latency minimization problem for satellite-assisted vehicle-to-vehicle networks by Reinforcement Learning. Under limited bandwidth, effective power and spectrum allocation schemes are necessary to overcome co-channel interference. However, the existing works [5]–[8] did not consider the impact of transmission power and spectrum allocation on the data transmission rate or latency for offloading tasks from terrestrial terminals to satellites. Moreover, existing algorithms suffer from issues such as limited applicability, poor generalization, and slow convergence. Large Language Models (LLMs) have emerged as a promising approach to solve these issues with their contextual learning and inference abilities, which have demonstrated outstanding optimization capability for wireless networks [9], [10].

In this paper, we aim to minimize the average latency of IoT terminals in a multi-satellite MEC network by optimizing the satellites' computation resource allocation and the IoT terminals' task offloading decisions, uplink sub-channel allocation, and transmission power allocation. Given that the formulated problem is non-convex and challenging to solve directly, we decompose it into two sub-problems: the computation resource allocation problem and the joint task offloading, power allocation, and sub-channel allocation problem. The former is proven to be convex and can be solved using the Lagrange multiplier method to obtain a closed-form result. For the latter, due to its complexity, traditional optimization algorithms often suffer from prolonged computing time [11], while existing AI algorithms typically require substantial time for model training or fine-tuning [12]. By harnessing the LLM's inference and generalization capabilities while avoiding the costs of dedicated model training, we propose an LLM-based optimizer that utilizes structured templates, pre-designed prompts, and an example pool to solve the second subproblem. An alternating optimization algorithm is devised based on the solutions to both sub-problems to solve the original problem. Simulation results are provided to evaluate the proposed algorithm for varying numbers of satellites and IoT terminals.

II. System Model and Problem Formulation

We consider a low-density satellite-MEC scenario in a remote area. The network contains M LEO satellites, denoted by $\mathcal{S} = \{S_1, S_2, \ldots, S_M\}$, which serve N IoT terminals sparsely distributed on the ground at fixed locations via I

Minghao Sun, Jinbo Hou, and Xiaoli Chu are with the School of Electrical and Electronic Engineering, the University of Sheffield, UK (e-mail:{msun39, jhou9, x.chu}@sheffield.ac.uk). Zitian Zhang is with the School of Information and Electronic Engineering, Zhejiang Gongshang University, China (e-mail: zitian.zhang@mail.zjgsu.edu.cn). Kehai Qiu is with the Department of Computer Science and Technology, University of Cambridge, UK (e-mail: kq218@cam.ac.uk), and Brunel University of London. Kezhi Wang is with the Department of Computer Science, Brunel University of London, UK(email: kezhi.wang@brunel.ac.uk). Xiaoli Chu and Zitian Zhang are co-corresponding authors.

This work was supported in part by the Horizon Europe Research and Innovation Program under grants 101086219 and 101086228, the UK EPSRC grants EP/X038971/1 and EP/Y028031/1, the Royal Society International Exchanges Award (IEC/NSFC/242607), Innovate UK COMET project (No: 10099265) and Royal Society Industry Fellowship (IF\R2\23200104).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TVT.2025.3612207, IEEE Transactions on Vehicular Technology

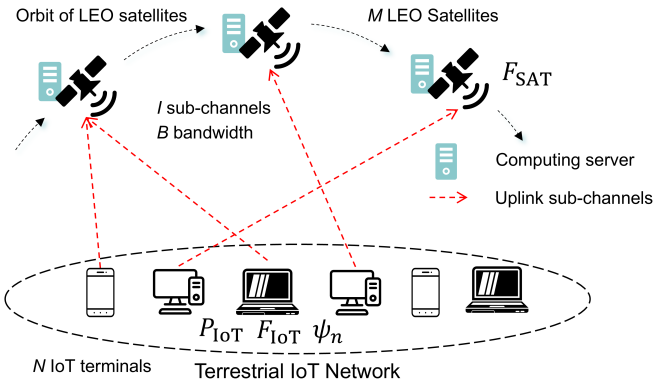

Fig. 1. Satellite Mobile Edge Computing Network.

orthogonal sub-channels, as shown in Fig. 1. Each satellite has a computational capacity F_{SAT} (in CPU cycle/s). The set of IoT terminals is represented by $\mathcal{T} = \{T_1, T_2, \ldots, T_N\}$. The n-th IoT terminal T_n has a task ψ_n with data size ε_n (in bits) and required computational density ρ (in CPU cycle/bit). Each terminal has a local computational capacity of F_{IoT} (in CPU cycle/s). An IoT terminal can either process its task locally or offload it to one of the satellites. The set of sub-channels is denoted by $\mathcal{A} = \{A_1, A_2, \ldots, A_I\}$. Each sub-channel has a bandwidth of B and can be employed by multiple IoT terminals simultaneously. One IoT terminal can use multiple sub-channels to send the task data to one satellite and can allocate various power across the sub-channels, with the total transmission power constrained by P_{IoT}. The scheduling period Δ is assumed to be sufficiently short such that the satellite-terminal geometry and channel fading coefficients can be regarded as quasi-static [11]. At the beginning of each scheduling period, the binary task offloading indicators $\alpha_{m,n}$ and sub-channel allocation indicators $\beta_{n,i}$ are updated by the system. Specifically, $\alpha_{m,n} = 1$ if task ψ_n is offloaded to satellite S_m, otherwise $\alpha_{m,n} = 0$; $\beta_{n,i} = 1$ if sub-channel A_i is allocated for transmitting task ψ_n, otherwise, $\beta_{n,i} = 0$.

A. Transmission Model

Since the distance from a satellite to an IoT terminal is much longer than that between any two IoT terminals, we assume that all the IoT terminals have approximately the same distance to the same satellite. The distance from an IoT terminal to satellite S_m is denoted by d_m and derived as [12]:

$$d_m = \sqrt{R_e^2 + (R_e + R_h)^2 - 2R_e(R_e + R_h)\cos\theta_m}, \quad (1)$$

$$\theta_m = \arccos\left(\frac{R_e}{R_e + R_h} \cdot \cos e_m\right) - e_m, \quad (2)$$

where R_e and R_h represent the radius of the Earth and the height of the satellite orbit above ground, respectively; θ_m is the geocentric angle of satellite S_m, and e_m is the elevation angle of S_m to an IoT terminal with lower limit $e_{\min}$ [13].

The channel gain of sub-channel A_i between satellite S_m and IoT terminal T_n is given by:

$$G^i_{m,n} = G_n\left(\frac{c}{4\pi f_{\mathrm{c}} d_m}\right)^2 (|h^i_{m,n}|)^2, \quad (3)$$

where G_n is the antenna gain (in dBi) of IoT terminal T_n, f_{c} is the carrier frequency, c denotes the speed of light, and $h^i_{m,n}$ represents the Rician fading with factor K.

If IoT terminal T_n offloads its task to satellite S_m, i.e., $\alpha_{m,n} = 1$, then the data rate from T_n to S_m is given by:

$$C_{m,n} = \sum_{i=1}^{I} B \log_2\left(1 + \frac{\alpha_{m,n}\beta_{n,i}P_{n,i}G^i_{m,n}}{\sum_{n'\neq n}^{N}\sum_{m'=1}^{M}\alpha_{m',n'}\beta_{n',i}P_{n',i}G^i_{m',n'} + N_0}\right), \quad (4)$$

where N_0 denotes the additive white Gaussian noise (AWGN) power at the satellite receiver, and $P_{n,i}$ denotes the transmission power from IoT terminal T_n in sub-channel A_i.

In the uplink, the transmission latency $\delta^{\mathrm{Trans}}_{m,n}$ of task ψ_n offloaded to satellite S_m is:

$$\delta^{\mathrm{Trans}}_{m,n} = \frac{\varepsilon_n}{C_{m,n}}. \quad (5)$$

The downlink transmission latency is neglected since the computation result size is typically much smaller than the uplink task data size [6].

B. Problem Formulation

If task ψ_n is processed locally, the computation time is:

$$\delta^{\mathrm{IoT}}_n = \frac{\varepsilon_n \rho}{F_{\mathrm{IoT}}}, \quad (6)$$

If task ψ_n is offloaded to satellite S_m, the computation time is:

$$\delta^{\mathrm{SAT}}_{m,n} = \frac{\varepsilon_n \rho}{F^{\mathrm{SAT}}_{m,n}}, \quad (7)$$

where $F^{\mathrm{SAT}}_{m,n}$ represents the computation resource (in CPU cycle/s) of satellite S_m allocated for processing task ψ_n.

The latency experienced by an IoT terminal includes the transmission latency and computation latency of its task. The latency of terminal T_n is given by:

$$\delta_n = \sum_{m=1}^{M} \alpha_{m,n}(\delta^{\mathrm{SAT}}_{m,n} + \delta^{\mathrm{Trans}}_{m,n}) + (1 - \sum_{m=1}^{M}\alpha_{m,n})\delta^{\mathrm{IoT}}_n. \quad (8)$$

To minimize the average latency of all IoT terminals, we formulate an optimization problem as follows,

$$\min_{\boldsymbol{P},\boldsymbol{F},\boldsymbol{\alpha},\boldsymbol{\beta}} \frac{1}{N}\sum_{n=1}^{N}\delta_n, \quad (9a)$$

$$\text{s.t.} \quad 0 \le \sum_{i=1}^{I}\beta_{n,i}P_{n,i} \le P_{\mathrm{IoT}}, \forall n \in \{1, ..., N\}, \quad (9b)$$

$$0 \le \sum_{n=1}^{N}\alpha_{m,n}F^{\mathrm{SAT}}_{m,n} \le F_{\mathrm{SAT}}, \forall m \in \{1, ..., M\}, \quad (9c)$$

$$\sum_{m=1}^{M}\alpha_{m,n} \le 1, \forall n \in \{1, ..., N\}, \quad (9d)$$

$$\delta_n \le \Delta, \forall n \in \{1, ..., N\}, \quad (9e)$$

$$\alpha_{m,n}, \beta_{n,i} \in \{0, 1\}, \quad (9f)$$

where $\boldsymbol{P} = \{P_{n,i} | n = 1, ..., N; i = 1, ..., I\}$, $\boldsymbol{F} = \{F^{SAT}_{m,n} | m = 1, ..., M; n = 1, ..., N\}$, $\boldsymbol{\alpha} = \{\alpha_{m,n} | m = 1, ..., M; n = 1, ..., N\}$, and $\boldsymbol{\beta} = \{\beta_{n,i} | n = 1, ..., N; i = 1, ..., I\}$. Constraint (9b) limits the total power allocated by

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TVT.2025.3612207, IEEE Transactions on Vehicular Technology

a terminal across all sub-channels; (9c) imposes the computational capacity of each satellite; (9d) ensures that a task can be offloaded to at most one satellite; (9e) ensures that a task must be processed within a scheduling period; and (9f) specifies the binary indicators.

III. Algorithm Design

The problem in (9) is non-convex due to the discrete solution space imposed by the binary variables [11]. To address this, it is divided into two sub-problems: the satellite computation resource allocation problem and the joint task offloading, power allocation, and sub-channel allocation problem. We first show that the former is convex and obtain a closed-form solution using the Lagrange multiplier method. Then, a novel LLM-based optimizer is proposed to solve the latter.

A. Satellite Computation Resource Allocation

For given values of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{P}$, problem (9) reduces to:

$$\begin{aligned} &\min_{\boldsymbol{F}} \frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M}\alpha_{m,n}\delta_{m,n}^{\mathrm{SAT}}, \\ &\text{s.t.} \quad (9\mathrm{c}). \end{aligned} \tag{10}$$

The second derivative of any element within the summation in (10) with respect to $F_{m,n}^{\mathrm{SAT}}$ is:

$$\frac{\partial^2(\alpha_{m,n}\delta_{m,n}^{\mathrm{SAT}})}{\partial {F_{m,n}^{\mathrm{SAT}}}^2} = \frac{2\alpha_{m,n}\varepsilon_n\rho}{{F_{m,n}^{\mathrm{SAT}}}^3} \geq 0, \tag{11}$$

where $\alpha_{m,n} \geq 0$, $\varepsilon_n > 0$, $\rho > 0$, and $F_{m,n}^{\mathrm{SAT}} > 0$. So the Hessian matrix of the objective function in (10) is a positive semi-definite matrix, and problem (10) is convex. The Lagrangian function $L(F_{m,n}^{\mathrm{SAT}}, \lambda_m)$ with Lagrange multiplier λ_m is as follows:

$$\begin{aligned} L(F_{m,n}^{\mathrm{SAT}}, \lambda_m) = &\frac{1}{N}(\sum_{m=1}^{M}\sum_{n=1}^{N}\alpha_{m,n}\delta_{m,n}^{\mathrm{SAT}}) \\ &+ \sum_{m=1}^{M}\lambda_m(\sum_{n=1}^{N}\alpha_{m,n}F_{m,n}^{\mathrm{SAT}} - F_{\mathrm{SAT}}). \end{aligned} \tag{12}$$

Taking the partial derivatives of $L(F_{m,n}^{\mathrm{SAT}}, \lambda_m)$ with respect to $F_{m,n}^{\mathrm{SAT}}$ and λ_m, and setting the results to zero, we have:

$$\begin{cases} \dfrac{\partial L(F_{m,n}^{\mathrm{SAT}}, \lambda_m)}{\partial F_{m,n}^{\mathrm{SAT}}} = -\dfrac{\alpha_{m,n}\varepsilon_n\rho}{{F_{m,n}^{\mathrm{SAT}}}^2} + \lambda_m\alpha_{m,n} = 0, \\ \dfrac{\partial L(F_{m,n}^{\mathrm{SAT}}, \lambda_m)}{\partial \lambda_m} = \sum\limits_{n=1}^{N}\alpha_{m,n}F_{m,n}^{\mathrm{SAT}} - F_{\mathrm{SAT}} = 0. \end{cases} \tag{13}$$

By solving the above equations, we obtain the optimal satellite computation resource allocation:

$$\tilde{F}_{m,n}^{\mathrm{SAT}} = \frac{F_{\mathrm{SAT}}\sqrt{\varepsilon_n\rho}}{\sum\limits_{k=1}^{N}\alpha_{m,k}\sqrt{\varepsilon_k\rho}}. \tag{14}$$

B. Joint task Offloading, Power Allocation and Sub-channel Allocation

For given $\boldsymbol{F}$, problem (9) reduces to :

$$\min_{\boldsymbol{P},\boldsymbol{\alpha}} \frac{1}{N}\sum_{n=1}^{N}\delta_n, \tag{15a}$$

$$\text{s.t.} \quad 0 \leq \sum_{i=1}^{I}P_{n,i} \leq P_{\mathrm{IoT}}, \forall n \in \{1, ..., N\}, \tag{15b}$$

$$(9\mathrm{c}), (9\mathrm{d}), (9\mathrm{f}),$$

where for simplicity, the sub-channel allocation indicators $\beta_{n,i}$ are omitted under the assumption that $\beta_{n,i} = 0$ if $P_{n,i} = 0$ and $\beta_{n,i} = 1$ if $P_{n,i} > 0$.

Problem (15) is still non-convex due to the binary constraint and fractional sum terms. To solve it, we propose an LLM-based optimizer as shown in Fig. 2 and detailed below.

The Generator Module consists of an LLM-based decision maker that uses prompts and an example pool as inputs to generate task offloading and power allocation solutions as outputs. The initial prompt includes the task description that outlines the objective based on (15), the environment description that details the system model with key parameters for customization, and the output format that specifies the template for generated solutions. The example pool contains an initial solution, i.e., the best solution to (15) among 100 randomly generated solutions, denoted by $\boldsymbol{\alpha_e}$, $\boldsymbol{P_e}$, and δ_e.

The Evaluation Module is composed of an LLM output extractor and a performance evaluation system. To avoid redundant texts due to hallucinations, the LLM extractor uses a task offloading extraction prompt and a power allocation extraction prompt to extract the intended solutions from the output text generated by the LLM. The performance of the extracted solutions $\boldsymbol{\alpha}$ and $\boldsymbol{P}$ is evaluated by substituting them into (15), and calculating the average latency $\delta = (\sum_{n=1}^{N}\delta_n)/N$; if any constraint of (15) is violated, $\delta = \infty$.

The LLM-based iterative algorithm begins by inputting the initial prompt and example pool into the LLM-based decision maker. The decision maker's output is then evaluated by the Evaluation Module, which compares δ with the average latency δ_e of the solution in the example pool and determines how the inputs to the decision maker should be updated in the next iteration as follows:

- **If** $\delta > \delta_e$**:** An expert prompt, based on domain-specific knowledge, will be input into the Generator Module.
- **If** $\delta < \delta_e$**:** The extracted solutions $\boldsymbol{\alpha}$ and $\boldsymbol{P}$ replace $\boldsymbol{\alpha_e}$ and $\boldsymbol{P_e}$ in the example pool, respectively.
- **If** $\delta = \delta_e$**:** If $\boldsymbol{\alpha} = \boldsymbol{\alpha_e}$ and $\boldsymbol{P} = \boldsymbol{P_e}$, a reminder prompt will be input into the Generator Module to prevent the LLM from being trapped in existing solutions; otherwise, the expert prompt will be input into the Generator Module.

The iteration process terminates when the example pool is not updated for ϵ consecutive iterations or when the number of iterations reaches a preset limit. The example pool returns the final solution to problem (15). Based on the solutions to the two sub-problems, an alternating optimization algorithm is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TVT.2025.3612207, IEEE Transactions on Vehicular Technology

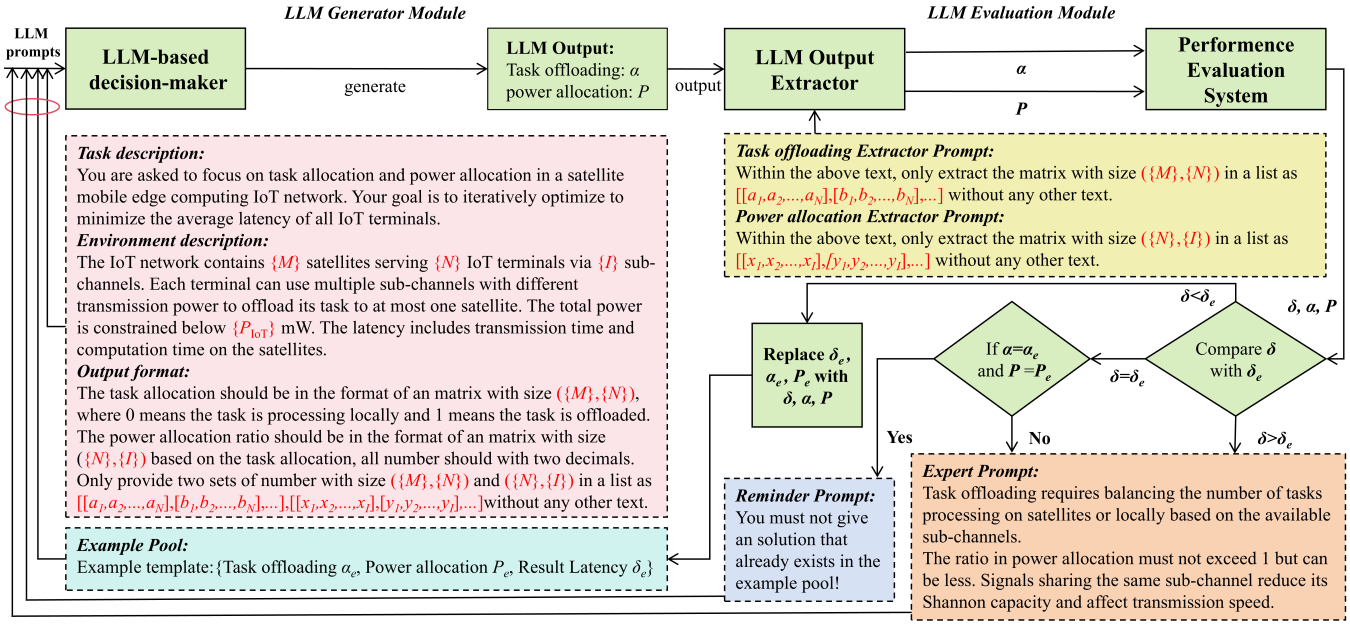

Fig. 2. The LLM-based framework for joint optimization of task offloading, power allocation, and sub-channel allocation.

Algorithm 1 LLM-Based Alternating Optimization

Input: M, N, I, ϵ, max_iterations

Initialize example pool $(\boldsymbol{\alpha}_e, \boldsymbol{P}_e, \delta_e)$ and initial prompt
Set iteration counter $i \leftarrow 0$, no_update_counter $\leftarrow 0$
$Prompt_i \leftarrow$ initial prompt
repeat
  Query LLM with $Prompt_i$ and extract $(\boldsymbol{\alpha}_{\text{new}}, \boldsymbol{P}_{\text{new}})$
  if any constraint of (15) is violated **then**
    Set $\delta_{\text{new}} \leftarrow \infty$
  else
    for $m = 1 : M$ and $n = 1 : N$ **do**
      $F_{m,n}^{\text{SAT}} \leftarrow$ (14)
    end for
    Compute average latency: $\delta_{\text{new}} \leftarrow$ (8) and (9)
  end if
  if $\delta_{\text{new}} < \delta_e$ **then**
    Update: $(\boldsymbol{\alpha}_e, \boldsymbol{P}_e, \delta_e) \leftarrow (\boldsymbol{\alpha}_{\text{new}}, \boldsymbol{P}_{\text{new}}, \delta_{\text{new}})$
    no_update_counter $\leftarrow 0$
  else if $(\boldsymbol{\alpha}_{\text{new}}, \boldsymbol{P}_{\text{new}}, \delta_{\text{new}}) = (\boldsymbol{\alpha}_e, \boldsymbol{P}_e, \delta_e)$ **then**
    $Prompt_{i+1} \leftarrow$ initial prompt::reminder prompt
    no_update_counter $\leftarrow$ no_update_counter $+1$
  else
    $Prompt_{i+1} \leftarrow$ initial prompt::expert prompt
    no_update_counter $\leftarrow$ no_update_counter $+1$
  end if
  $i \leftarrow i + 1$
until no_update_counter $\geq \epsilon$ or $i \geq$ max_iterations
return $\boldsymbol{\alpha}_e, \boldsymbol{P}_e, \mathbf{F}, \delta_e$

devised to solve (9). The pseudo-code is shown in Algorithm 1.

Since most existing commercial LLM APIs are memoryless, both task offloading and power allocation solutions must be generated within a single conversation (i.e., a series of prompt-response exchanges) with the LLM. The example pool can help maintain continuity across iterations. Some open-source localized models like Llama 3 can mitigate this issue, but currently lag behind in inference performance.

IV. Simulation Results

This section presents the performance evaluation of the proposed LLM-based alternating optimization algorithm (LLM), building on (14) and the LLM-based optimizer in Fig. 2. It is well recognized that employing commercial LLM APIs introduces additional network latency due to data exchanges with cloud servers, whereas local deployment of LLMs entails significant hardware and maintenance costs. Therefore, commercial APIs are particularly suitable for edge devices with limited computational resources, offering access to high-performance LLMs at relatively low costs. Conversely, local deployment ensures greater data confidentiality and, by interacting directly with local devices, significantly reduces communication overhead, making it preferable in latency-sensitive and/or safety-critical scenarios. Given the substantial hardware requirements for deploying high-performance LLMs locally, this study utilizes different LLM models through commercial APIs for performance evaluation. Nevertheless, our proposed algorithm is compatible with local deployment-based LLMs too. By testing three widely used models, GPT-4o, LLaMA-3.1-70B, and DeepSeek-R1-0528 under the same prompt design and algorithmic framework, we observe that only GPT-4o successfully converged to an optimized solution, where both the LLaMA-3.1-70B and DeepSeek-R1-0528 failed to complete the algorithm due to hallucinations. The LLaMA-3.1-70B model sometimes returned wrong dimensions of decision matrices, while the DeepSeek-R1-0528 model often generated non-binary values of α within the first 5 iterations, both failing to complete the algorithm. Therefore, GPT-4o is adopted in the following simulations due

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI: 10.1109/TVT.2025.3612207, IEEE Transactions on Vehicular Technology

to its superior reliability and reasoning performance across iterative optimization tasks. The benchmarks for performance comparison include the GA, where the initial population size is set at 400 with a crossover probability of 0.5 and a mutation probability of 0.2; the Deep Deterministic Policy Gradient (DDPG) algorithm with a greedy exploration strategy, where the exploration rate is 0.05, and both the Actor and Critic networks contain three layers with the learning rate of 10^{-5}; Random Choice (RC), where each task has an equal probability of being processed locally or offloaded to one of the satellites, and the other optimization variables are uniformly distributed within their allowed value ranges; Processing All Locally (PAL), where each IoT terminal processes its task locally and all the variables in problem (9) are set to zero. In the simulation, the task data size ϵ_n is uniformly distributed between 0.3 MB and 0.6 MB, and other parameters are shown in Table I unless otherwise specified.

TABLE I
SIMULATION PARAMETERS [6], [12], [13]

Notation	Value	Notation	Value	Notation	Value
N_0	-134dBm	ρ_n	100cycle/bit	P_{IoT}	0.2mW
Δ	1s	F_{IoT}	0.5Gcycle/s	$e_{\min}$	10°
B	10MHz	F_{SAT}	20Gcycle/s	R_e	6371km
G_n	3dBi	f_{c}	3.49GHz	R_h	550km

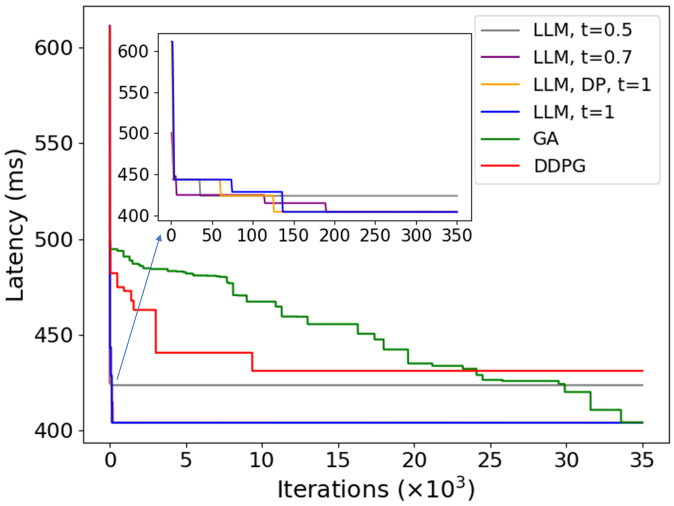

Fig. 3. Convergence of the LLM-based algorithm for different temperature t and different prompt (DP) wording, GA, and DDPG for $M = 3, N = 10, I = 4$.

Fig. 3 shows the convergence performance of the proposed LLM-based algorithm for different values of temperature t, which is a hyperparameter that controls the sharpness of the LLM's output probability distribution, and for different prompt (DP) wording, in comparison with GA and DDPG. Higher temperature values lead to more diverse and stochastic outputs. Our simulations tested temperature values of 0.5, 0.7, 1, 1.3, and 1.5 and found that for t = 1.3 and 1.5, the LLM-based algorithm failed to complete due to occasional dimension mismatches in the generated solutions. For t = 0.5, 0.7, and 1, the LLM-based algorithms converge within 40 to 200 iterations, significantly faster than both DDPG and GA. The results indicate that t = 1 offers the best balance between convergence speed and latency minimization performance. Therefore, in the subsequent experiments, the LLM-based algorithm adopts t = 1. Additionally, the performance of the LLM-based algorithm with different prompt wording is close to that with the prompt wording defined in Fig. 2, suggesting that the LLM-based algorithm is not sensitive to the wording of prompts as long as the underlying intent remains consistent.

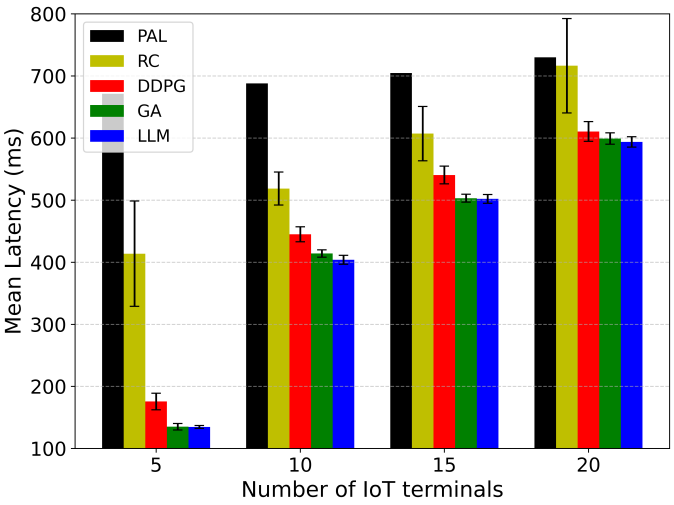

Fig. 4. Mean latency and standard deviation vs. the number of IoT terminals for $M = 3, I = 4$.

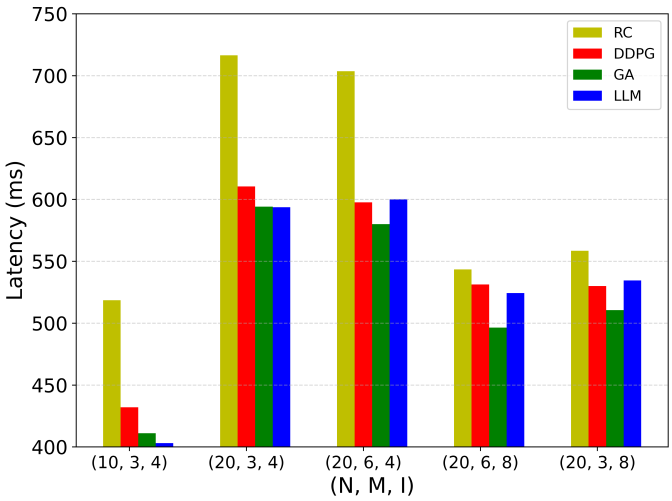

Fig. 5. Average latency for different values of (N, M, I).

Fig. 4 shows the mean latency and standard deviation achieved by different algorithms across 10 experimental runs for 5, 10, 15, and 20 IoT terminals served by 3 satellites. It shows that as the number of terminals increases, the average latency rises for all the considered schemes. This is mainly due to increased co-channel interference. It also shows that the proposed LLM-based algorithm achieves the lowest mean latency and smallest standard deviation for every considered number of IoT terminals, followed by GA and DDPG. Although DDPG has been widely used for solving non-convex problems, it is prone to being stuck in local optima and is highly sensitive

to hyperparameter settings. In contrast, the proposed LLM-based algorithm encourages broader solution exploration when repetitive solutions are detected and avoids getting stuck in local optima through prompt-based adaptations.

Fig. 5 shows the average latency achieved by four schemes for different values of (N, M, I). The proposed LLM-based algorithm consistently outperforms RC across all scenarios. For the scenarios of (20, 6, 4), (20, 6, 8), and (20, 3, 8), as the decision matrices contain significantly more float numbers, increasing the likelihood of hallucinations or invalid outputs, the LLM-based algorithm is outperformed by GA.

V. Conclusions and Future Work

In this work, we have proposed a novel LLM optimizer combined with the Lagrange multiplier method to minimize the average latency of IoT terminals in a multi-satellite MEC network. Simulation results demonstrate that the LLM-based alternating optimization algorithm converges significantly faster than both GA and DDPG while obtaining a lower average latency for the IoT terminals. The LLM-based algorithm exhibits strong adaptability and effectively achieves the optimization objective across varying numbers of IoT terminals and satellites. Our results also show that for high-dimensional scenarios, it would be necessary to adopt a multi-agent approach—a promising direction for future work. In addition, we plan to investigate other practical issues in satellite MEC networks, such as the response time of LLMs. It is mainly determined by the latency of generating the first token and the subsequent time required to produce the full output text, which scales with the number of output tokens [14].

References

[1] F. Guo, et al., "Enabling massive IoT toward 6G: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 11 891–11 915, Aug. 2021.
[2] W.-C. Chien, et al., "Heterogeneous space and terrestrial integrated networks for IoT: Architecture and challenges," *IEEE Network*, vol. 33, no. 1, pp. 15–21, Feb. 2019.
[3] M. De Sanctis, et al., "Satellite communications supporting internet of remote things," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 113–123, Feb. 2016.
[4] K. Zhang, et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, Aug. 2016.
[5] X. Gao, et al., "Hierarchical dynamic resource allocation for computation offloading in LEO satellite networks," *IEEE Internet of Things Journal*, vol. 11, no. 11, pp. 19 470–19 484, Feb. 2024.
[6] L. Zhao, et al., "QoS-aware multi-hop task offloading in satellite-terrestrial edge networks," *IEEE Internet of Things Journal*, vol. 11, no. 19, pp. 31 453–31 466, Jun. 2024.
[7] S. Zhang, et al., "Joint computing and communication resource allocation for satellite communication networks with edge computing," *China Communications*, vol. 18, no. 7, pp. 236–252, Jul. 2021.
[8] G. Cui, et al., "Joint offloading and resource allocation for satellite assisted vehicle-to-vehicle communication," *IEEE Systems Journal*, vol. 15, no. 3, pp. 3958–3969, Sep. 2020.
[9] S. Xu, et al., "Large multi-modal models (LMMs) as universal foundation models for AI-native wireless systems," *IEEE Network*, vol. 38, no. 5, pp. 10–20, Jul. 2024.
[10] Q. Li, et al., "Latency-aware generative semantic communications with pre-trained diffusion models," *IEEE Wireless Communications Letters*, vol. 13, no. 10, pp. 2652–2656, Jul. 2024.
[11] Y. Zhao, et al., "Partial computation offloading in satellite-based three-tier cloud-edge integration networks," *IEEE Transactions on Wireless Communications*, vol. 23, no. 2, pp. 836–847, Jun. 2024.
[12] Z. Zhao, et al., "Federated deep recurrent Q-learning for task partitioning and resource allocation in satellite mobile edge computing assisted industrial IoT," *IEEE Internet of Things Journal*, vol. 11, no. 15, pp. 26 444–26 458, May 2024.
[13] J.M. Gongora-Torres, et al., "Link budget analysis for LEO satellites based on the statistics of the elevation angle," *IEEE Access*, vol. 10, pp. 14 518–14 528, Jan. 2022.
[14] H. Zhou, et al., "Generative AI as a service in 6G edge-cloud: Generation task offloading by in-context learning," *IEEE Wireless Communications Letters*, vol. 14, no. 3, pp. 711–715, Mar. 2025.