Resource Allocation for Stable LLM Training in Mobile Edge Computing

Chang Liu
Graduate College
Nanyang Technological University
Singapore
liuc0063@e.ntu.edu.sg

Jun Zhao
College of Computing and Data Science
Nanyang Technological University
Singapore
junzhao@ntu.edu.sg

ABSTRACT

As mobile devices increasingly become focal points for advanced applications, edge computing presents a viable solution to their inherent computational limitations, particularly in deploying large language models (LLMs). However, despite the advancements in edge computing, significant challenges remain in efficient training and deploying LLMs due to the computational demands and data privacy concerns associated with these models. This paper explores a collaborative training framework that integrates mobile users with edge servers to optimize resource allocation, thereby enhancing both performance and efficiency. Our approach leverages parameter-efficient fine-tuning (PEFT) methods, allowing mobile users to adjust the initial layers of the LLM while edge servers handle the more demanding latter layers. Specifically, we formulate a multi-objective optimization problem to minimize the total energy consumption and delay during training. We also address the common issue of instability in model performance by incorporating stability enhancements into our objective function. Through novel fractional programming technique, we achieve a stationary point for the formulated problem. Simulations demonstrate that our method reduces the energy consumption as well as the latency, and increases the reliability of LLMs across various mobile settings.

CCS CONCEPTS

• **Networks → Network resources allocation**.

KEYWORDS

Mobile edge computing, large language model, wireless networks.

Corresponding author: Jun Zhao
Chang Liu is a PhD student supervised by Jun Zhao.

1 INTRODUCTION

The advent of large language models (LLMs) marks a significant milestone in the advancement of artificial intelligence and offers unparalleled capabilities in natural language processing, generation, and understanding. The desire for ubiquitous access to Artificial intelligence (AI) capabilities has driven a significant trend and demand toward the deployment and even training of these computationally intensive models directly on mobile devices [2, 18, 30]. Users seek real-time, personalized experiences and decision-making support across a diverse array of applications, from healthcare to customer service, which only such advanced models can provide. Additionally, there is a growing emphasis on decentralization in computing to enhance privacy and data security by processing sensitive information locally on the device, rather than transmitting it to distant data centers. However, this aspiration faces a tough challenge: the substantial computational resources required by LLMs. These models necessitate sophisticated hardware configurations that far exceed the capabilities of standard mobile devices [19].

Mobile edge computing (MEC) emerges as a transformative solution to this challenge by bringing computational resources closer to the data source [26]. MEC enables data processing at the network's edge, utilizing distributed computing resources geographically proximate to where the data originates and is consumed. Augmented with LLMs, mobile edge servers can process and understand complex queries locally, minimizing the need for constant communication with centralized cloud infrastructure. This not only improves response times but also enhances privacy and data security by processing sensitive information closer to its source.

However, we still face challenges in how LLMs can be optimized in real-time mobile applications to achieve the best possible performance. The challenges involve tailoring these models in resource-constrained environments. One solution is in-context learning. It provides the LLM with a few examples of the desired task within the input prompt, allowing the model to adapt its behavior based on these examples without changing its parameters. But the effectiveness of in-context learning is constrained by the model's context window size, and it doesn't lead to persistent improvements in the model's capabilities. Moreover, recent studies have shown that in-context learning may struggle with reliability and hallucination [13]. Alternatively, many studies propose parameter-efficient fine-tuning (PEFT) methods [5, 12, 17]. These methods have been demonstrated to yield state-of-the-art performance in model training and optimization, significantly reducing the computational overhead traditionally associated with such processes. Inspired by this body of work, we propose a novel scenario for collaboratively training LLMs, harnessing the combined capabilities

of mobile users and edge servers. In this proposed model, mobile users are responsible for fine-tuning the first several layers of the LLM, capitalizing on the parameter-efficient techniques that require less computational power and are thus more suitable for mobile environments. At the same time, the edge servers undertake the task of training the remaining layers, leveraging their processing capabilities to manage the more resource-intensive aspects of the training process.

Nevertheless, it has been found by many works that the prevalent fine-tuning methods are afflicted with instability concerns [9, 21, 34]. The fine-tuning approach, partitioning the computation process between users and edge servers, inherently introduces potential variances in the model's learning dynamics. When the initial several layers of a large language model are tailored to the idiosyncrasies of local data, these layers may start to generate data representations that are highly specialized or customized to the user's local context. Such misalignment can manifest as model instability, where small variations in local data could result in disproportionately large changes in the model's output, reducing its reliability and robustness in real-world applications. Thus, addressing model stability is essential. A stable model ensures minor changes in the training process don't lead to large performance discrepancies, making the model robust and reliable across various conditions and data distributions. This goal is even more critical in our proposed collaborative training framework, where the training workload is divided between mobile devices and edge servers. To solve this problem, we propose to incorporate model stability as a component of our objective function. This approach aims to reduce performance variance across training instances, ensuring that the fine-tuning process yields consistently high-quality results, regardless of the minor fluctuations inherent in distributed training environments. The contributions of this paper are summarized as follows:

- We introduce a collaborative training framework that combines mobile users and edge servers. This framework leverages PEFT methods, allowing mobile users to adjust the initial layers of the LLM while edge servers handle the more demanding latter layers.
- We formulate a multi-objective optimization problem that aims to concurrently minimize total energy consumption and user-experienced delay. At the same time, we enhance the stability of LLMs by integrating model stability considerations into our optimization objectives.
- To quantify the relationship between the number of the fine-tuned layers and the model stability, we provide the upper bound of the average-replace-one stability through theoretical analysis.
- To address the multi-objective optimization problem, we divide the problem into two parts. In the first part, we optimize the offloading decisions and resource allocation through the application of a novel fractional programming technique, which could find a stationary point with local or global optimal guarantee. For the second part, the Concave-Convex Procedure (CCCP) is employed to optimize the user-to-edge association problem.

The structure of this paper is laid out as follows: Section 2 reviews the literature and works related to our study. Section 3 outlines the architecture of the MEC-based LLM framework. Following that, Section 4 details the analytical exploration. The outcomes of our simulations are presented in Section 5. We conclude the paper with Section 6.

2 RELATED WORK

In this section, we review the existing literature related to our work.

Resource allocation in mobile edge computing. In [20], the authors propose a Lyapunov optimization-based dynamic computation offloading algorithm to optimize the execution delay and the task-dropping cost in MEC system. When addressing the offloading decision problem, they apply an exhaustive search strategy, assessing the objective values across three discrete options to determine the optimal solution. Dinh *et al.* [6] propose to minimize the execution latency and the user devices' energy consumption in MEC. They use an exhaustive search approach and a semidefinite relaxation (SDR)-based approach to optimize the CPU frequency. However, the exhaustive search approach is not practical in implementation due to its high complexity, and the SDR-based approach has no global or local optimality guarantee. In [3], Chen *et al.* optimize the computation resource allocated to each task to minimize the computation and communication delay. To handle the multiplication of two decision variables (i.e., the computation resource allocation and the offloading decision), they adopt alternative optimization (AO) techniques. Xu *et al.* [31] formulate a cooperative resource optimization problem to optimize the offloading decision and resource allocation in vehicular edge computing. Yet, they decouple the resource allocation variables from the offloading decision variable, and then use a deep reinforcement learning-based approach to solve it. Zhan *et al.* [32] optimize the computation offloading scheduling and resource allocation in unmanned aerial vehicle (UAV)-enabled MEC system. They propose a two-stage alternating optimization algorithm to optimize the offloading scheduling, resource allocation and time duration alternatively.

In contrast, Wang *et al.* [29] obtain the optimal solution in a semi-closed form for offloading decisions and resource allocation in MEC with wireless power transfer. However, their study exclusively focuses on minimizing the total energy consumption without integrating delay considerations into the objective function. Consequently, while it facilitates the determination of the optimal CPU frequency, it inherently simplifies the selection process to the minimal processing unit frequency that meets the latency requirements. Nonetheless, in this paper, we incorporate delay considerations into the objective function, thereby introducing a higher level of complexity to the solution process for resource allocation. As a result, the optimal solution to our problem cannot be directly ascertained. In Table 1, a comparative analysis is presented between this paper and the aforementioned related works.

PEFT vs. In-Context Learning (ICL). Recent studies have demonstrated the superiority of PEFT methods over ICL in various scenarios. Mosbach *et al.* [22] conduct a fair comparison of ICL and fine-tuning approaches across different tasks and model sizes. They find that fine-tuning outperforms in-context learning across different performance metrics. Liu *et al.* [16] also rigorously demonstrate that PEFT surpasses ICL in both accuracy and computational efficiency.

The model stability of fine-tuned large language models. Extensive efforts have focused on developing algorithms aimed at improving the stability of the fine-tuning process. Based on the idea of dropout, Lee *et al.* [14] present "Mixout" regularization technique to selectively combine the parameters of two pre-trained

Table 1: A comparative overview of this paper and prior works on MEC.

| Reference	Objective Function		Optimization technique used to solve the multiplication of variables
	Energy incorporated	Delay incorporated	
Mao *et al.* [20]	✗	✓	Exhaustive search-based strategy
Chen *et al.* [3]	✗	✓	Alternative optimization
Xu *et al.* [31]	✗	✓	Deep reinforcement learning-based approach
Zhan *et al.* [32]	✓	✗	Alternating optimization
Wang *et al.* [29]	✓	✗	Lagrange duality method
This paper	✓	✓	Novel fractional programming technique
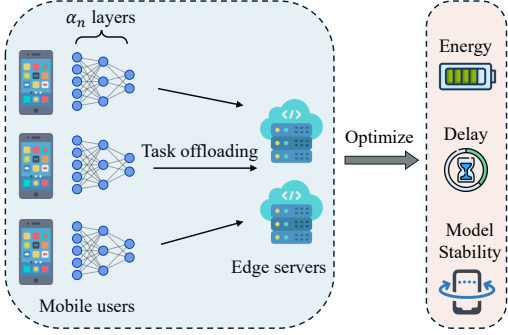
Figure 1: The proposed system model consists of N mobile users and M edge servers. Our optimization problem aims to minimize energy consumption and delay while improving the LLM stability.

language models. This approach effectively regularizes the learning process, improving the stability of the model. Houlsby *et al.* [12] propose a transfer method based on a bottleneck adapter architecture. He *et al.* [11] conduct a comprehensive comparison between two PEFT methods: fine-tuning and adapter-based tuning. Their works demonstrate that selectively tuning a subset of the parameters from pre-trained models contributes to enhancing the stability of the model.

For the stability analysis, Fu *et al.* [7] harmonize the array of PEFT strategies by framing them within the paradigm of sparse fine-tuning models. They provide a theoretical analysis that highlights sparsity's function as a regulatory mechanism for the original model, effectively imposing a constraint on the upper limit of model stability. However, their reliance on pointwise hypothesis stability to evaluate model stability focuses on the sensitivity of individual predictions to changes in the training data. In contrast, our work employs the average-replace-one stability measure which assesses the model's overall performance variation when a single training instance is replaced. In edge computing, we focus more on maintaining high levels of service reliability and efficiency across the entire network, rather than optimizing the outcome for individual predictions. Average-replace-one stability aligns with this objective by providing a macroscopic view of model stability.

3 SYSTEM MODEL

In this section, we first present the system model, including local computation model, edge computation model and LLM stability. After that, we formulate the multi-objective optimization problem. We consider an MEC system consisting of N users and M edge servers, as described in Figure 1. Assume all the users in the system

train LLMs with the same architecture. Let Υ be the total number of transformer layers in the LLM. User n fine-tunes the first α_n layers locally, after which the intermediate results are sent to a certain edge server to complete the remaining training process. Let d_n denote the length of input tokens of user n for training. For the energy and delay calculation for training LLMs, we follow the setting in [15]. Let $\psi(d_n)$ be the FLOPs per token required to train one transformer layer, $\psi(d_n) = 72Bd_n h^2 + 12Bd_n^2 h$ where B is the batch size and h is the dimensionality of the hidden states.

3.1 Local Computation Model

When user n is training one transformer layer locally, the delay for computation can be given by:

$$T_n^{cmp} = \frac{\psi(d_n)}{f_n C_n^U D_n^U}, \tag{1}$$

where f_n is the GPU frequency of user n, C_n^U is the number of cores of the GPU at user n and D_n^U is the number of FLOPs per cycle per core of the GPU. The relationship between the GPU's power consumption and its clock speed is cubic, i.e., power $= \kappa_1 f_n^3$. Here, κ_1 is the coefficient reflecting the power usage per cubic cycle per second ([in Watt/(cycle/s)^3]), dependent on the specific GPU architecture. Hence, when training one transformer layer, the energy expenditure for local computations is established as follows:

$$E_n^{cmp} = \kappa_1 f_n^3 \times T_n^{cmp} = \frac{\kappa_1 f_n^2 \psi(d_n)}{C_n^U D_n^U}. \tag{2}$$

Upon completing local computations, users transmit the intermediate results to edge servers for further processing. The association between users and edge servers is represented by $\chi_{n,m}$ with $\chi_{n,m} = 1$ signifying that user n has selected edge server m for further computations, and $\chi_{n,m} = 0$ indicating no such association. In this context, we adopt Frequency-Division Multiple Access (FDMA) such that communications between users and edge servers are free from mutual interference. The power used for transmission by user n is denoted as p_n. Following the principles of the Shannon-Hartley theorem [4], the transmission rate between user n and edge server m can be formulated as $r_{n,m} = b_{n,m} \log_2(1 + \frac{g_{n,m}p_n}{\sigma^2 b_{n,m}})$, where σ^2 represents the power of the noise , $b_{n,m}$ denotes the bandwidth that edge server m assigned to user n, p_n is the transmission power of user n, and $g_{n,m}$ is the channel gain between user n and edge server m. Let $s(d_n)$ be the size of the intermediate results for user n. Therefore, the energy consumption of wireless transmission for user n is:

$$E_n^{com} = \sum_{m \in \mathcal{M}} \chi_{n,m} \frac{s(d_n)p_n}{r_{n,m}}. \tag{3}$$

When user n is training the first α_n layers locally, the computation of both time and energy expenditure is:

$$Cost_n^u = \alpha_n \cdot (\omega_t T_n^{cmp} + \omega_e E_n^{cmp}) + \omega_e E_n^{com}. \tag{4}$$

Here, ω_t serves as the weighting and normalization factor, reflecting the priority given to minimizing delay, while ω_e represents the

weighting and normalization factor that underscores the importance of reducing energy consumption.

3.2 Edge Computation Model

When edge server m trains one transformer layer for user n, the time taken for the computation can be expressed as follows:

$$T_{n,m}^{cmp} = \frac{\psi(d_n)}{f_{n,m} C_m^E D_m^E}, \quad (5)$$

where $f_{n,m}$ denotes the GPU frequency of edge server m assigned to user n, and C_m^E represents the total core count of the GPU within edge server m, and D_m^E signifies the computational capability of each core, measured in floating point operations per cycle, for the GPU located at edge server m. Thus, the energy required by edge server m to train one transformer layer for user n can be quantified as follows:

$$E_{n,m}^{cmp} = \frac{\kappa_2 f_{n,m}^2 \psi(d_n)}{C_m^E D_m^E}, \quad (6)$$

where κ_2 is a coefficient that varies based on the architecture of the chip. The energy used for downlink transmission from the edge servers to the users is not considered in this calculation, due to the substantially higher power capacities of the edge servers compared to the users. Furthermore, in comparison to the energy requirements for training the LLM, the energy expended on transmission by the edge servers is considered negligible.

Since there are Υ transformer layers in total, $(\Upsilon - \alpha_n)$ layers are processed at the corresponding edge server. As a result, the incurred cost for conducting the training tasks for users at edge server m is calculated by integrating both the time delays and energy expenditures into a weighted sum:

$$Cost_m^E = \sum_{n \in \mathcal{N}} \chi_{n,m} (\Upsilon - \alpha_n)(\omega_t T_{n,m}^{cmp} + \omega_e E_{n,m}^{cmp}). \quad (7)$$

3.3 LLM Stability

In this paper, we use the Average-replace-one Stability (AS) proposed by [25] to measure the mode stability. AS is a measure of how much an individual prediction is affected by small changes in the training dataset. It serves as a crucial metric for ensuring that our fine-tuned language model remains consistent and reliable, despite the variability in local data from user to user. Next, we give the definition of the average-replace-one stability.

DEFINITION 1 (AVERAGE-REPLACE-ONE STABILITY). *Given a loss function ℓ and training dataset $\mathcal{S} = \{z_1, \ldots, z_k\}$, an algorithm $\mathcal{A}$ demonstrates the average-replace-one stability (AS) with a bound β if the following condition is met: $\forall i \in \{1, \ldots, k\}$,*

$$\mathbb{E}_{\mathcal{S}} \left[\left| \ell(\mathcal{A}(\mathcal{S}), z_i) - \ell(\mathcal{A}(\mathcal{S}^i), z_i) \right| \right] \leq \beta, \quad (8)$$

where $\mathcal{A}(\mathcal{S})$ denotes the model obtained after the algorithm $\mathcal{A}$ has been trained on the dataset $\mathcal{S}$, and $\ell(\mathcal{A}(\mathcal{S}), z_i)$ is the loss function evaluated at a particular data point z_i using the model given by $\mathcal{A}(\mathcal{S})$. $\mathcal{S}^i$ represents the training dataset with the i-th sample replaced with z_i', i.e., $\mathcal{S}^i = \{z_1, \ldots, z_{i-1}, z_i', \ldots, z_k\}$.

This definition implies that for every individual element z_i in a dataset of size k, the expected disparity in the loss computed by algorithm $\mathcal{A}$ when trained with the complete dataset versus the dataset lacking that specific sample is bounded by β.

3.4 Problem Formulation

With the computation and communication model above, we then formulate the joint optimization problem that aims to minimize the system's cost while minimizing the Average-replace-one Stability (AS) of the LLMs, by optimizing the following variables: the number of transformer layers that execute locally: $\boldsymbol{\alpha} := [\alpha_n|_{n \in \mathcal{N}}]$,

the user-to-edge server association: $\boldsymbol{\chi} := [\chi_{n,m}|_{n \in \mathcal{N}, m \in \mathcal{M}}]$, the transmission power of the users: $\boldsymbol{p} := [p_n|_{n \in \mathcal{N}}]$, the bandwidth allocation: $\boldsymbol{b} := [b_{n,m}|_{n \in \mathcal{N}, m \in \mathcal{M}}]$, the users' GPU frequency: $\boldsymbol{f}^U := [f_n|_{n \in \mathcal{N}}]$ and the edge servers' GPU frequency allocation: $\boldsymbol{f}^E := [f_{n,m}|_{n \in \mathcal{N}, m \in \mathcal{M}}]$. Similar to delay and energy, we also give a weighting and normalization parameter ω_s to the AS. The joint optimization problem is formulated as follows:

$$\text{Problem } \mathbb{P}_1 : \min_{\boldsymbol{\alpha}, \boldsymbol{\chi}, \boldsymbol{p}, \boldsymbol{b}, \boldsymbol{f}^U, \boldsymbol{f}^E} \sum_{n \in \mathcal{N}} Cost_n^u + \sum_{m \in \mathcal{M}} Cost_m^E + \omega_s AS, \quad (9)$$

$$\text{s.t. } \alpha_n \in \{1, 2, \ldots, \Upsilon\}, \forall n \in \mathcal{N}, \quad (9a)$$

$$\chi_{n,m} \in \{0, 1\}, \forall n \in \mathcal{N}, m \in \mathcal{M}, \quad (9b)$$

$$\sum_{m \in \mathcal{M}} \chi_{n,m} = 1, \forall n \in \mathcal{N}, \quad (9c)$$

$$p_n \leq p_n^{max}, \forall n \in \mathcal{N}, \quad (9d)$$

$$\sum_{n \in \mathcal{N}} \chi_{n,m} b_{n,m} = b_m^{max}, \forall m \in \mathcal{M}, \quad (9e)$$

$$f_n \leq f_n^{max}, \forall n \in \mathcal{N}, \quad (9f)$$

$$\sum_{n \in \mathcal{N}} \chi_{n,m} f_{n,m} = f_m^{max}, \forall m \in \mathcal{M}. \quad (9g)$$

Given the inherent challenges in quantifying the average-replace-one stability, we commence by presenting the following theorem to facilitate addressing the optimization problem. We assume the loss function $\ell(\cdot)$ is L-Lipschitz and strong convex. These two assumptions are widely employed in the analysis of the behavior of neural networks [23, 24].

THEOREM 1. *If a user fine-tunes a proportion α of the parameters, the expectation of the loss has an AS bounded by $\frac{2L^2}{k(1-\alpha)}$. I.e., $\forall i \in \{1, \ldots, k\}$,*

$$\mathbb{E}_{\mathcal{S}} \left[\left| \ell(\mathcal{A}(\mathcal{S}), z_i) - \ell(\mathcal{A}(\mathcal{S}^i), z_i) \right| \right] \leq \frac{2L^2}{k(1-\alpha)}. \quad (10)$$

PROOF. The proof can be found in Appendix A. □

Theorem 1 provides a quantifiable measure of model stability and bridges the concept of "model stability" with a measurable quantity. Since the "model stability" term is not quantitative in problem $\mathbb{P}_1$, we re-formulate problem $\mathbb{P}_1$ into the following $\mathbb{P}_2$ by replacing the sum of AS with the sum of the AS's upper bound of all the users:

$$\text{Problem } \mathbb{P}_2 : \min_{\boldsymbol{\alpha}, \boldsymbol{\chi}, \boldsymbol{p}, \boldsymbol{b}, \boldsymbol{f}^U, \boldsymbol{f}^E} \sum_{n \in \mathcal{N}} Cost_n^u + \sum_{m \in \mathcal{M}} Cost_m^E + \omega_s \sum_{n \in \mathcal{N}} \frac{2L^2}{k_n(1 - \frac{\alpha_n}{\Upsilon})}, \quad (11)$$

$$\text{s.t. } (9a) - (9g).$$

While the optimal solutions to problem $\mathbb{P}_1$ and $\mathbb{P}_2$ may not be strictly equivalent in a mathematical sense, $\mathbb{P}_2$ serves as a practical approximation of $\mathbb{P}_1$. By using the upper bound from Theorem 1, we are optimizing for the worst-case scenario of model instability. Problem $\mathbb{P}_2$ falls into the category of Mixed Integer Nonlinear Programming (MINLP) problem. This classification arises due to the inclusion of both integer-valued decision variables and nonlinear terms involving products of variables, a combination that inherently induces non-convexity in the problem space. The non-convex nature of this problem makes it especially challenging to solve because it cannot be addressed using standard optimization methods, which typically rely on the problem being convex. In order to tackle the non-convex problem, we optimize $\boldsymbol{\alpha}, \boldsymbol{p}, \boldsymbol{b}, \boldsymbol{f}^U, \boldsymbol{f}^E$ and $\boldsymbol{\chi}$ iteratively. Specifically, in the first step, we fix $\boldsymbol{\chi}$ and utilize a novel fractional programming technique motivated by Zhao *et al.* [33] to optimize $\boldsymbol{\alpha}, \boldsymbol{p}, \boldsymbol{b}, \boldsymbol{f}^U, \boldsymbol{f}^E$ by transforming the non-convex problem into a series of parametric convex problems. In the second step, given $\boldsymbol{\alpha}, \boldsymbol{p}, \boldsymbol{b}, \boldsymbol{f}^U, \boldsymbol{f}^E$, the method of CCCP is adopted to facilitate the solution to $\boldsymbol{\chi}$ by solving a sequence of convex problems.

4 PROPOSED ALGORITHM

In this section, we provide a detailed solution to the optimization problem.

4.1 Optimizing α, p, b, f^U, f^E given χ

The discrete variable α_n is difficult to handle. Thus, we first relax α_n to continuous variables, which will be rounded back to the nearest integer later. For problem $\mathbb{P}_2$, to optimize α, p, b, f^U, f^E given χ means to solve the following optimization problem:

Problem $\mathbb{P}_3(\chi)$: $\min\limits_{\alpha,p,b,f^U,f^E} H(\alpha,p,b,f^U,f^E) =$

$$\sum_{n\in\mathcal{N}} Cost_n^u + \sum_{m\in\mathcal{M}} Cost_m^E + \omega_s \sum_{n\in\mathcal{N}} \frac{2L^2}{k_n\cdot(1-\frac{\alpha_n}{\Upsilon})}, \quad (12)$$

s.t. $1 \leq \alpha_n \leq \Upsilon, \forall n \in \mathcal{N}$, (12a)

(9d)–(9g).

Problem $\mathbb{P}_3$ involves fraction term and multiplication terms, which makes it difficult to solve using standard optimization algorithms. Motivated by the novel fractional programming technique proposed in [33], we next transform problem $\mathbb{P}_3$ into a series of $\mathbb{P}_4$:

Problem $\mathbb{P}_4(\chi,z,v,q)$: $\min\limits_{\alpha,p,b,f^U,f^E} K(\alpha,p,b,f^U,f^E,z,v,q) =$

$$\sum_{n\in\mathcal{N}} \left(\alpha_n^2 z_n + \frac{(\omega_t \frac{\psi(d_n)}{f_n C_n^U D_n^U} + \omega_e \frac{\kappa_1 f_n^2 \psi(d_n)}{C_n^U D_n^U})^2}{4z_n} \right) +$$

$$\omega_e \sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}} \chi_{n,m}\left((p_n d_n)^2 v_{n,m} + \frac{1}{4r_{n,m}^2 v_{n,m}} \right) +$$

$$\sum_{n\in\mathcal{N}}\sum_{m\in\mathcal{M}} \chi_{n,m}\left((\Upsilon-\alpha_n)^2 q_{n,m} + \frac{\left(\omega_t \frac{\psi(d_n)}{f_{n,m} C_m^E D_m^E} + \omega_e \frac{\kappa_2 f_{n,m}^2 \psi(d_n)}{C_m^E D_m^E}\right)^2}{4q_{n,m}} \right)$$

$$+ \omega_s \sum_{n\in\mathcal{N}} \frac{2L^2}{k_n\cdot(1-\frac{\alpha_n}{\Upsilon})}, \quad (13)$$

s.t. (12a), (9d)–(9g),

where the auxiliary variables $z := [z_1, z_2, \ldots, z_n]$ with $z_n > 0$, $v := [v_{1,1}, v_{1,2}, \ldots, v_{1,m}, \ldots, v_{n,m}]$ with $v_{n,m} > 0$ and $q := [q_{1,1}, q_{1,2}, \ldots, q_{1,m}, \ldots, q_{n,m}]$ with $q_{n,m} > 0$.

Problem $\mathbb{P}_3$ involves non-convex terms and is difficult to handle. Therefore, we formulate the above problem $\mathbb{P}_4$. Next, we introduce an AO algorithm for problem $\mathbb{P}_4$. After that, we propose Proposition 1 to explain how we can tackle problem $\mathbb{P}_3$ through a series of convex problem $\mathbb{P}_4$ instances.

First, we introduce the AO algorithm for problem $\mathbb{P}_4$. Overall, we alternatively optimize α, p, b, f^U, f^E and z, v, q. Specifically, we begin with an initial feasible $[\alpha^{(0)}, p^{(0)}, b^{(0)}, f^{U(0)}, f^{E(0)}]$. Next, we denote $A(f_n)$ and $B(f_{n,m})$ as:

$$A(f_n) = \omega_t \frac{\psi(d_n)}{f_n C_n^U D_n^U} + \omega_e \frac{\kappa_1 f_n^2 \psi(d_n)}{C_n^U D_n^U}, \quad (14)$$

$$B(f_{n,m}) = \omega_t \frac{\psi(d_n)}{f_{n,m} C_m^E D_m^E} + \omega_e \frac{\kappa_2 f_{n,m}^2 \psi(d_n)}{C_m^E D_m^E}. \quad (15)$$

We assign $z_n^{(0)}$ to be $\frac{A(f_n^{(0)})}{2\alpha_n^{(0)}}$, which is the optimal value of z_n when optimizing $\alpha_n^2 z_n + \frac{A^2(f_n)}{4z_n}$ with respect to z_n, while keeping α_n, f_n fixed at $\alpha_n^{(0)}$, $f_n^{(0)}$; we assign $v_{n,m}^{(0)}$ to be $\frac{1}{2p_n^{(0)} d_n^{(0)} r_{n,m}}$, which is the optimal value of $v_{n,m}$ when optimizing $(p_n d_n)^2 v_{n,m} + \frac{1}{4r_{n,m}^2 v_{n,m}}$ with respect to $v_{n,m}$, while keeping p_n, d_n fixed at $p_n^{(0)}$, $d_n^{(0)}$; we assign $q_{n,m}^{(0)}$ to be $\frac{B(f_{n,m}^{(0)})}{2(\Upsilon-\alpha_n^{(0)})}$, which is the optimal value of $q_{n,m}$

when optimizing $(\Upsilon-\alpha_n)^2 q_{n,m} + \frac{B^2(f_{n,m})}{4q_{n,m}}$ with respect to $q_{n,m}$, while keeping α_n, $f_{n,m}$ fixed at $\alpha_n^{(0)}$, $f_{n,m}^{(0)}$.

After that, we solve problem $\mathbb{P}_4(\chi, z^{(0)}, v^{(0)}, q^{(0)})$, from which we derive the solution $[\alpha^{(1)}, p^{(1)}, b^{(1)}, f^{U(1)}, f^{E(1)}]$, and subsequently update $[z^{(1)}, v^{(1)}, q^{(1)}]$. This procedure is repeated in an iterative fashion: during the $(t+1)$-th iteration, we set $z_n^{(t)}$, $v_{n,m}^{(t)}$ and $q_{n,m}^{(t)}$ as $\frac{A(f_n^{(t)})}{2\alpha_n^{(t)}}$, $\frac{1}{2p_n^{(t)} d_n^{(t)} r_{n,m}}$ and $\frac{B(f_{n,m}^{(t)})}{2(\Upsilon-\alpha_n^{(t)})}$, and then solve $\mathbb{P}_4(\chi, z^{(t)}, v^{(t)}, q^{(t)})$, to obtain $[\alpha^{(t+1)}, p^{(t+1)}, b^{(t+1)}, f^{U(t+1)}, f^{E(t+1)}]$. The above AO process converges when the difference between the objective function of problem $\mathbb{P}_4(\chi, z^{(t-1)}, v^{(t-1)}, q^{(t-1)})$ and problem $\mathbb{P}_4(\chi, z^{(t)}, v^{(t)}, q^{(t)})$ falls below a predefined small error tolerance. Then, we propose the following proposition to explain how we solve $\mathbb{P}_3$ through the AO process for $\mathbb{P}_4$.

PROPOSITION 1. *We can derive a stationary point for problem $\mathbb{P}_3$ by applying the AO process outlined above for problem $\mathbb{P}_4$ until convergence.*

PROOF. Denote "α, p, b, f^U, f^E" as "$\bigstar$" and "z, v, q" as "$\blacklozenge$". In the first step in the AO process, we optimize $\blacklozenge$ while keeping $\bigstar$ fixed, i.e., letting $z_n^\# = \frac{A(f_n)}{2\alpha_n}$, $v_{n,m}^\# = \frac{1}{2p_n d_n r_{n,m}}$, $q_{n,m}^\# = \frac{B(f_{n,m})}{2(\Upsilon-\alpha_n)}$. When we substitute back $z_n^\#$, $v_{n,m}^\#$, $q_{n,m}^\#$ to $K(\bigstar, \blacklozenge)$, $K(\bigstar, \blacklozenge)$ will become $H(\bigstar)$, i.e.,

$$K(\bigstar, \blacklozenge)|_{z_n=z_n^\#, v_{n,m}=v_{n,m}^\#, q_{n,m}=q_{n,m}^\#} = H(\bigstar). \quad (16)$$

Next, we investigate the partial derivative of $K(\bigstar, \blacklozenge)$ w.r.t $\bigstar$:

$$\frac{\partial K(\bigstar, \blacklozenge)}{\partial \alpha_n} = \frac{2L^2\omega_s}{k_n\Upsilon\cdot(1-\frac{\alpha_n}{\Upsilon})^2} + 2z_n\alpha_n - 2\sum_{m\in\mathcal{M}} \chi_{n,m}q_{n,m}\cdot(\Upsilon-\alpha_n), \quad (17)$$

$$\frac{\partial K(\bigstar, \blacklozenge)}{\partial p_n} = \sum_{m\in\mathcal{M}} \chi_{n,m}\omega_e\cdot$$
$$\left(2d_n^2 v_{n,m}p_n - \frac{\ln^2(2)\, g_{n,m}}{2b_{n,m}^3 v_{n,m}\sigma^2\cdot\left(\frac{g_{n,m}p_n}{b_{n,m}\sigma^2}+1\right)\ln^3\left(\frac{g_{n,m}p_n}{b_{n,m}\sigma^2}+1\right)} \right) \quad (18)$$

$$\frac{\partial K(\bigstar, \blacklozenge)}{\partial b_{n,m}} = \frac{\chi_{n,m}\omega_e \ln^2(2)}{2v_{n,m}\ln^2\left(\frac{g_{n,m}p_n}{\sigma^2 b_{n,m}}+1\right)b_{n,m}^3}\cdot$$
$$\left(\frac{g_{n,m}p_n}{\sigma^2\ln\left(\frac{g_{n,m}p_n}{\sigma^2 b_{n,m}}+1\right)\cdot\left(\frac{g_{n,m}p_n}{\sigma^2 b_{n,m}}+1\right)b_{n,m}} - 1 \right), \quad (19)$$

$$\frac{\partial K(\bigstar, \blacklozenge)}{\partial f_n} = \frac{\psi^2(d_n)\cdot(\kappa_1\omega_e f_n^3 + \omega_t)\cdot(2\kappa_1\omega_e f_n^3 - \omega_t)}{2z_n f_n^3\cdot(C_n^U D_n^U)^2}, \quad (20)$$

$$\frac{\partial K(\bigstar, \blacklozenge)}{\partial f_{n,m}} = \frac{\chi_{n,m}\psi^2(d_n)\cdot(\kappa_2\omega_e f_{n,m}^3 + \omega_t)\cdot(2\kappa_2\omega_e f_{n,m}^3 - \omega_t)}{2q_{n,m}f_{n,m}^3\cdot(C_m^E D_m^E)^2}. \quad (21)$$

From (17) to (21), it can be found that

$$\left(\frac{\partial K(\bigstar, \blacklozenge)}{\partial \alpha_n}\right)|_{z_n=\frac{A(f_n)}{2\alpha_n}, q_{n,m}=\frac{B(f_{n,m})}{2(\Upsilon-\alpha_n)}} =$$
$$\frac{2L^2\omega_s}{k_n\Upsilon\cdot(1-\frac{\alpha_n}{\Upsilon})^2} + A(f_n) - \sum_{m\in\mathcal{M}} \chi_{n,m}\cdot B(f_{n,m}), \quad (22)$$

$$\left(\frac{\partial K(\bigstar, \blacklozenge)}{\partial p_n}\right)|_{v_{n,m}=\frac{1}{2p_n d_n r_{n,m}}} = \sum_{m\in\mathcal{M}} \chi_{n,m}\omega_e\cdot$$
$$\left(\frac{d_n}{b_{n,m}\log_2(1+\frac{g_{n,m}p_n}{\sigma^2 b_{n,m}})} - \frac{\ln(2)g_{n,m}p_n d_n}{b_{n,m}(g_{n,m}p_n+b_{n,m}\sigma^2)\ln^2\left(\frac{g_{n,m}p_n}{b_{n,m}\sigma^2}+1\right)} \right), \quad (23)$$

$$\left(\frac{\partial K(\bigstar, \blacklozenge)}{\partial b_{n,m}}\right)|_{v_{n,m}=\frac{1}{2p_n d_n r_{n,m}}} = \frac{\chi_{n,m}\omega_e \ln(2)p_n d_n}{b_{n,m}^2\ln\left(\frac{g_{n,m}p_n}{b_{n,m}\sigma^2}+1\right)}.$$

$$\left(\frac{g_{n,m}p_n}{\sigma^2 \ln\left(\frac{g_{n,m}p_n}{\sigma^2 b_{n,m}}+1\right)\cdot\left(\frac{g_{n,m}p_n}{\sigma^2 b_{n,m}}+1\right)b_{n,m}} - 1 \right), \tag{24}$$

$$\left(\frac{\partial K(\bigstar,\blacklozenge)}{\partial f_n} \right)\big|_{z_n=\frac{A(f_n)}{2\alpha_n}} = \frac{\alpha_n\psi(d_n)\cdot(2\kappa_1\omega_e f_n^3-\omega_t)}{f_n^2 C_n^V D_n^V}, \tag{25}$$

$$\left(\frac{\partial K(\bigstar,\blacklozenge)}{\partial f_{n,m}} \right)\big|_{q_{n,m}=\frac{B(f_{n,m})}{2(\Upsilon-\alpha_n)}} = \frac{(\Upsilon-\alpha_n)\chi_{n,m}\psi(d_n)\cdot(2\kappa_2\omega_e f_{n,m}^3-\omega_t)}{f_{n,m}^2 C_m^R D_m^R}. \tag{26}$$

Besides, the partial derivative of $H(\bigstar)$ is given by:

$$\frac{\partial H(\bigstar)}{\partial \alpha_n} = \omega_t\frac{\psi(d_n)}{f_n C_n^U D_n^U} + \omega_e\frac{\kappa_1 f_n^2\psi(d_n)}{C_n^U D_n^U} + \frac{2L^2\omega_s}{k_n\Upsilon\cdot\left(1-\frac{\alpha_n}{\Upsilon}\right)^2} -$$
$$\sum_{m\in\mathcal{M}}\chi_{n,m}(\omega_t\frac{\psi(d_n)}{f_{n,m}C_m^E D_m^E}+\omega_e\frac{\kappa_2 f_{n,m}^2\psi(d_n)}{C_m^E D_m^E}), \tag{27}$$

$$\frac{\partial H(\bigstar)}{\partial p_n} = \sum_{m\in\mathcal{M}}\ln(2)\,\omega_e\chi_{n,m}d_n\cdot$$
$$\frac{(g_{n,m}p_n+b_{n,m}\sigma^2)\ln(\frac{g_{n,m}p_n}{b_{n,m}\sigma^2}+1)-g_{n,m}p_n}{b_{n,m}\cdot(g_{n,m}p_n+b_{n,m}\sigma^2)\ln^2(\frac{g_{n,m}p_n}{b_{n,m}\sigma^2}+1)}, \tag{28}$$

$$\frac{\partial H(\bigstar)}{\partial b_{n,m}} = \frac{\omega_e\ln(2)\,\chi_{n,m}d_n g_{n,m}p_n^2}{\sigma^2\ln^2\left(\frac{g_{n,m}p_n}{\sigma^2 b_{n,m}}+1\right)\left(\frac{g_{n,m}p_n}{\sigma^2 b_{n,m}}+1\right)b_{n,m}} - \frac{\omega_e\ln(2)\,\chi_{n,m}d_n p_n}{\ln\left(\frac{g_{n,m}p_n}{\sigma^2 b_{n,m}}+1\right)b_{n,m}^2}, \tag{29}$$

$$\frac{\partial H(\bigstar)}{\partial f_n} = -\frac{\alpha_n\omega_t\psi(d_n)}{C_n^V D_n^V f_n^2} + \frac{2\alpha_n\omega_e\kappa_1 f_n\psi(d_n)}{C_n^V D_n^V}, \tag{30}$$

$$\frac{\partial H(\bigstar)}{\partial f_{n,m}} = \chi_{n,m}(\Upsilon-\alpha_n)\left(\frac{2\omega_e\kappa_2 f_{n,m}\psi(d_n)}{C_m^R D_m^R} - \frac{\omega_t\psi(d_n)}{C_m^R D_m^R f_{n,m}^2}\right). \tag{31}$$

From (22)–(26) and (27)–(31), it can be observed that:

$$\frac{\partial H(\bigstar)}{\partial \alpha_n} = \left(\frac{\partial K(\bigstar,\blacklozenge)}{\partial \alpha_n}\right)\big|_{z_n=\frac{A(f_n)}{2\alpha_n},q_{n,m}=\frac{B(f_{n,m})}{2(\Upsilon-\alpha_n)}}, \tag{32a}$$

$$\frac{\partial H(\bigstar)}{\partial p_n} = \left(\frac{\partial K(\bigstar,\blacklozenge)}{\partial p_n}\right)\big|_{v_{n,m}=\frac{1}{2p_n d_n r_{n,m}}}, \tag{32b}$$

$$\frac{\partial H(\bigstar)}{\partial b_{n,m}} = \left(\frac{\partial K(\bigstar,\blacklozenge)}{\partial b_{n,m}}\right)\big|_{v_{n,m}=\frac{1}{2p_n d_n r_{n,m}}}, \tag{32c}$$

$$\frac{\partial H(\bigstar)}{\partial f_n} = \left(\frac{\partial K(\bigstar,\blacklozenge)}{\partial f_n}\right)\big|_{z_n=\frac{A(f_n)}{2\alpha_n}}, \tag{32d}$$

$$\frac{\partial H(\bigstar)}{\partial f_{n,m}} = \left(\frac{\partial K(\bigstar,\blacklozenge)}{\partial f_{n,m}}\right)\big|_{q_{n,m}=\frac{B(f_{n,m})}{2(\Upsilon-\alpha_n)}}. \tag{32e}$$

The process of AO in minimizing $K(\bigstar,\blacklozenge)$ is non-increasing. To elaborate, it's observed that $K(\bigstar^{(i)},\blacklozenge^{(i)}) \leq K(\bigstar^{(i-1)},\blacklozenge^{(i)}) \leq K(\bigstar^{(i-1)},\blacklozenge^{(i-1)})$. Thus, $K(\bigstar^{(i)},\blacklozenge^{(i)})$ tends towards convergence as the iteration count i increases infinitely. Assume after the AO process, $\bigstar$ and $\blacklozenge$ converges to $(\bigstar^*,\blacklozenge^*)$. It indicates: ① $\blacklozenge^*$ is the optimal value of $\blacklozenge$ if we fix $\bigstar$ as $\bigstar^*$ and minimize $K(\bigstar^*,\blacklozenge)$, and ② $\bigstar^*$ is the optimial value of $\bigstar$ if we fix $\blacklozenge$ as $\blacklozenge^*$ and minimize $K(\bigstar,\blacklozenge^*)$.

From result ②, we know that $\bigstar^*$ satisfies the Karush–Kuhn–Tucker (KKT) conditions of problem $\mathbb{P}_4$. With γ,μ denoting the Lagrange multipliers for the inequality constraints and equality constraints, respectively, the Lagrangian function of problem $\mathbb{P}_4$ is given by:

$$L_{\mathbb{P}_4}(\bigstar,\blacklozenge,\gamma,\mu) = K(\bigstar,\blacklozenge) + \sum_{n\in\mathcal{N}}\Big(\gamma_{1,n}(1-\alpha_n)+$$
$$\gamma_{2,n}(\alpha_n-\Upsilon)+\gamma_{3,n}(p_n-p_n^{max})+\gamma_{4,n}(f_n-f_n^{max})\Big)+$$
$$\sum_{m\in\mathcal{M}}\Big(\mu_{1,m}(\sum_{n\in\mathcal{N}}\chi_{n,m}b_{n,m}-b_m^{max})+\mu_{2,m}(\sum_{n\in\mathcal{N}}\chi_{n,m}f_{n,m}-f_m^{max})\Big). \tag{33}$$

The KKT conditions of problem $\mathbb{P}_4$ is given by:

Stationarity:
$$\frac{\partial L_{\mathbb{P}_4}}{\partial\alpha^*} = \frac{\partial L_{\mathbb{P}_4}}{\partial p^*} = \frac{\partial L_{\mathbb{P}_4}}{\partial b^*} = \frac{\partial L_{\mathbb{P}_4}}{\partial f^{U*}} = \frac{\partial L_{\mathbb{P}_4}}{\partial f^{E*}} = 0, \tag{34}$$
Primal feasibility:
$$\bigstar^* \text{ satisfy (12a), (9d)} - \text{(9g)}, \tag{35}$$
Dual feasibility:
$$\gamma_{1,n},\gamma_{2,n},\gamma_{3,n},\gamma_{4,n} \geq 0, \forall n\in\mathcal{N}, \tag{36}$$
Complementary slackness:
$$\gamma_{1,n}(1-\alpha_n^*)=0, \gamma_{3,n}(p_n^*-p_n^{max})=0,$$
$$\gamma_{2,n}(\alpha_n^*-\Upsilon)=0, \gamma_{4,n}(f_n^*-f_n^{max})=0, \forall n\in\mathcal{N}. \tag{37}$$

We rewrite $L_{\mathbb{P}_4}$ as $L_{\mathbb{P}_4} = K(\bigstar,\blacklozenge)+Q(\bigstar,\gamma,\mu)$. Then, (34) can be rewritten as:

$$\frac{\partial L_{\mathbb{P}_4}}{\partial\alpha} = \frac{\partial K(\bigstar^*,\blacklozenge^*)}{\partial\alpha} + \frac{\partial Q(\bigstar^*,\gamma,\mu)}{\partial\alpha} = 0, \tag{38a}$$

$$\frac{\partial L_{\mathbb{P}_4}}{\partial p} = \frac{\partial K(\bigstar^*,\blacklozenge^*)}{\partial p} + \frac{\partial Q(\bigstar^*,\gamma,\mu)}{\partial p} = 0, \tag{38b}$$

$$\frac{\partial L_{\mathbb{P}_4}}{\partial b} = \frac{\partial K(\bigstar^*,\blacklozenge^*)}{\partial b} + \frac{\partial Q(\bigstar^*,\gamma,\mu)}{\partial b} = 0, \tag{38c}$$

$$\frac{\partial L_{\mathbb{P}_4}}{\partial f^U} = \frac{\partial K(\bigstar^*,\blacklozenge^*)}{\partial f^U} + \frac{\partial Q(\bigstar^*,\gamma,\mu)}{\partial f^U} = 0, \tag{38d}$$

$$\frac{\partial L_{\mathbb{P}_4}}{\partial f^E} = \frac{\partial K(\bigstar^*,\blacklozenge^*)}{\partial f^E} + \frac{\partial Q(\bigstar^*,\gamma,\mu)}{\partial f^E} = 0. \tag{38e}$$

Substituting (32a) – (32e) into (38a) – (38e), we can obtain:

$$\frac{\partial H(\bigstar^*)}{\partial\alpha} + \frac{\partial Q(\bigstar^*,\gamma,\mu)}{\partial\alpha} = 0, \tag{39a}$$

$$\frac{\partial H(\bigstar^*)}{\partial p} + \frac{\partial Q(\bigstar^*,\gamma,\mu)}{\partial p} = 0, \tag{39b}$$

$$\frac{\partial H(\bigstar^*)}{\partial b} + \frac{\partial Q(\bigstar^*,\gamma,\mu)}{\partial b} = 0, \tag{39c}$$

$$\frac{\partial H(\bigstar^*)}{\partial f^U} + \frac{\partial Q(\bigstar^*,\gamma,\mu)}{\partial f^U} = 0, \tag{39d}$$

$$\frac{\partial H(\bigstar^*)}{\partial f^E} + \frac{\partial Q(\bigstar^*,\gamma,\mu)}{\partial f^E} = 0. \tag{39e}$$

At the same time, with γ,μ denoting the Lagrange multipliers, the Lagrangian function of problem $\mathbb{P}_3$ can be given by:

$$L_{\mathbb{P}_3}(\bigstar,\blacklozenge,\gamma,\mu) = H(\bigstar)+Q(\bigstar,\gamma,\mu). \tag{40}$$

Therefore, (39a) – (39e) indicate that

$$\frac{\partial L_{\mathbb{P}_3}}{\partial\alpha} = \frac{\partial L_{\mathbb{P}_3}}{\partial p} = \frac{\partial L_{\mathbb{P}_3}}{\partial b} = \frac{\partial L_{\mathbb{P}_3}}{\partial f^U} = \frac{\partial L_{\mathbb{P}_3}}{\partial f^E} = 0. \tag{41}$$

Then, (34) – (37) are equivalent to:

Stationarity:
$$\frac{\partial L_{\mathbb{P}_3}}{\partial\alpha} = \frac{\partial L_{\mathbb{P}_3}}{\partial p} = \frac{\partial L_{\mathbb{P}_3}}{\partial b} = \frac{\partial L_{\mathbb{P}_3}}{\partial f^U} = \frac{\partial L_{\mathbb{P}_3}}{\partial f^E} = 0, \tag{42}$$
Primal feasibility:
$$\bigstar^* \text{ satisfy (12a), (9d)} - \text{(9g)}, \tag{43}$$
Dual feasibility:
$$\gamma_{1,n},\gamma_{2,n},\gamma_{3,n},\gamma_{4,n} \geq 0, \forall n\in\mathcal{N}, \tag{44}$$
Complementary slackness:
$$\gamma_{1,n}(1-\alpha_n^*)=0, \gamma_{3,n}(p_n^*-p_n^{max})=0,$$
$$\gamma_{2,n}(\alpha_n^*-\Upsilon)=0, \gamma_{4,n}(f_n^*-f_n^{max})=0, \forall n\in\mathcal{N}. \tag{45}$$

The above (42) – (45) indicate that $\bigstar^*$ is a stationary point for problem $\mathbb{P}_3$. Therefore, the proof is concluded. □

It can be easily verified that $\mathbb{P}_4$ is a convex optimization problem and can be solved by utilizing convex optimization solvers such as CVX [8]. According to Proposition 1, we are able to obtain a stationary point for $\mathbb{P}_3$ after solving $\mathbb{P}_4$.

4.2 Optimizing χ given α, p, b, f^U, f^E

Firstly, to reduce the computational complexity, we convert discrete variables into continuous ones. Without loss of equivalence, constraint (9b) can be reformulated as:

$$\begin{cases} \chi_{n,m} \in [0,1], \ \forall n \in \mathcal{N}, \ m \in \mathcal{M}, & (46) \\ \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}} \chi_{n,m} \cdot (1 - \chi_{n,m}) \leq 0. & (47) \end{cases}$$

By replacing constraint (9b) with constraints (46) and (47), the discrete variables are transformed into continuous ones, thereby reducing the computation complexity of the problem.

With fixed α, p, b, f^U, f^E, solving problem $\mathbb{P}_1$ is equivalent to solving the following problem:

Problem $\mathbb{P}_5(\alpha, p, b, f^U, f^E)$:

$$\min_{\chi} \sum_{n \in \mathcal{N}} Cost_n^u + \sum_{m \in \mathcal{M}} Cost_m^E, \qquad (48)$$

s.t. (9c), (9e), (9g), (46), (47).

However, constraint (47) remains a non-convex constraint. Thus, further measures are required to efficiently tackle this challenge. Next, we convert problem $\mathbb{P}_5$ into an equivalent problem that has linear constraints, which we then address using the CCCP method. To this end, we introduce the following lemma:

LEMMA 1. Let $G(\chi_{n,m}) = \sum_{n \in \mathcal{N}} Cost_n^u + \sum_{m \in \mathcal{M}} Cost_m^E$. With any $\chi_{n,m}^0$ satisfying (9c), (9e), (9g), and (46), for all $\varrho > \varrho_0$ where

$$\varrho_0 = \frac{G(\chi_{n,m}^0) - \min\{G(\chi_{n,m}): (9c), (9e), (9g), (46)\}}{\min\{\sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}} \chi_{n,m}(1-\chi_{n,m}): (9c), (9e), (9g), (46)\}}, \tag{49}$$

problem $\mathbb{P}_5$ has the same optimal solution with problem $\mathbb{P}_6$, which is defined as follows:

Problem $\mathbb{P}_6(\alpha, p, b, f^U, f^E)$:

$$\min_{\chi} \sum_{n \in \mathcal{N}} Cost_n^u + \sum_{m \in \mathcal{M}} Cost_m^E + \varrho \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}} \chi_{n,m}(1-\chi_{n,m}), \quad (50)$$

s.t. (9c), (9e), (9g), (46).

It is worth noting that problem $\mathbb{P}_6$ is derived from problem $\mathbb{P}_5$ by integrating the concave constraint (47) into the objective function as a penalization term.

PROOF. The proof can be directly derived from Theorem 1 in [1]. □

Problem $\mathbb{P}_6$ involves subtracting a quadratic convex function from a linear function, while its constraints are linear in nature. According to [27], problem $\mathbb{P}_6$ falls under the category of indefinite quadratic problem, which is a subset of the broader class of problems known as the difference of convex problems. With the objective function of problem $\mathbb{P}_6$ being differentiable, we can effectively address problem $\mathbb{P}_6$ using the CCCP method, which involves employing a first-order Taylor series approximation [10] to refine

$\sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}} \chi_{n,m}(\chi_{n,m} - 1)$. Specifically, it updates the expression to:

$$\sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}} \chi_{n,m}^{(i)}(\chi_{n,m}^{(i)} - 1) + \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}} (2\chi_{n,m}^{(i)} - 1)(\chi_{n,m} - \chi_{n,m}^{(i)}), \quad (51)$$

where $\chi_{n,m}^{(i)}$ indicates the value of $\chi_{n,m}$ at the i-th iteration. After that, the CCCP method systematically approaches resolution by iteratively engaging in a sequence of linear problems. The CCCP method not only simplifies complex issues by breaking them down into more manageable linear tasks but also ensures a structured progression towards finding an optimal solution through successive approximations. However, directly solving problem $\mathbb{P}_6$ to reach a feasible solution for problem $\mathbb{P}_5$ might not always be viable. To navigate this challenge, our approach involves generating several local optimum solutions for problem $\mathbb{P}_6$. This is achieved by applying the CCCP algorithm multiple times, initiating each iteration from a different feasible starting point specific to problem $\mathbb{P}_6$. The optimal solution is then determined by selecting the one that presents the smallest average value among these.

5 SIMULATIONS

In this section, we present the performance of the proposed approach through simulations. The simulated MEC network has 50 mobile users and 10 edge servers by default. Assume the users and edge servers collaboratively train Meta's open-source large language model Meta-AI (LLaMA-7B) which consists of 32 transformer layers [28]. The path loss is modeled as $128.1 + 37.6\log(\texttt{distance})$ and Gaussian noise power is $\sigma^2 = -134$dBm. The maximum transmission power p_n^{max} for the users is set in the range of 1 to 2 W. The maximum GPU frequency f_n^{max} for users and f_m^{max} for edge servers are chosen from [0.5,1] and [1,3] respectively. The total bandwidth b_m^{max} for each edge server is 20 MHz.

For LLM training, the batch size B is set to 512 and the dimensionality of the hidden states h is set to 1024. The lengths of input tokens for each user are randomly generated from 512 to 1024. We assume the mobile users are equipped with mobile devices with GPU such as Apple A15, whose GPU has 4 to 6 cores. Thus, the number of cores of the GPU at the user side C_n^U is chosen between 4 to 6. The number of FLOPs per cycle per core D_n^U is all set to 1. The edge servers are presumed to be equipped with advanced GPUs such as NVIDIA Tesla T4 and NVIDIA Tesla V100, therefore the number of cores of the GPU C_m^E is randomly assigned values from the interval [2560, 5120]. The number of FLOPs per cycle per core D_m^E is chosen between 1 and 2.

5.1 The performance of the proposed collaborative training method

In Figure 2, we present an analysis of system performance across three computing approaches: the proposed collaborative training method, edge server training method and local training method. Figure 2 (a) illustrates the energy consumption associated with each approach, where the proposed collaborative method demonstrates a balanced reduction in energy usage compared to the edge server training and local training methods. Figure 2 (b) depicts the delay experienced under each approach, showing that the proposed method effectively minimizes delay, achieving a performance closer
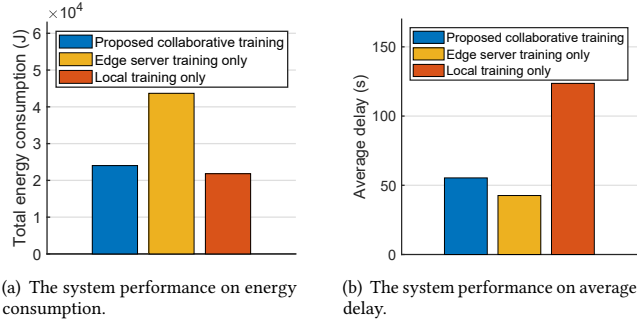
(a) The system performance on energy consumption.

(b) The system performance on average delay.

Figure 2: Comparison of system performance with and without the proposed collaborative training approach.

to the edge server training approach while significantly outperforming the local training method. These results demonstrate the efficiency and effectiveness of the proposed collaborative training method in optimizing both energy consumption and system delay.

5.2 The performance of proposed algorithms under weighting factors

Next, we compare the performance of the proposed method when the weighting factors for energy, delay and model stability vary. The default weighting factors after normalization are all set to 1 for energy, delay and model stability. A larger weighting factor denotes enhanced prioritization of system attributes such as energy efficiency, latency, or model stability. The three additional methodologies employed for comparative analysis with our proposed method are listed as follows:

- **Alternating optimization:** This method is the most commonly employed strategy in the related literature as discussed in Section 2. It systematically alternates between optimizing offloading decisions and the allocation of computational or communication resources.
- **Optimize α only:** This approach solely focuses on optimizing the offloading decision α, while implementing a random strategy for resource allocation.
- **Optimize resource only:** This method concentrates exclusively on the optimization of resource allocation, while employing a random approach to the offloading decision α.

In Figure 3, we adjust the weighting factors for energy, delay and model stability from 1 to 10, respectively. For each setting where one attribute's weighting factor varied from 1 to 10, the weighting factors for the other two attributes are held constant at 1. In Figure 3 (a), the proposed methodology consistently attains the lowest energy consumption among the methods evaluated. The alternating optimization approach secures the second-best performance. Conversely, the method that solely optimizes α exhibits the poorest results. This suboptimal performance can be attributed to the fact that the α-only optimization method neglects resource allocation considerations, which are crucial in minimizing energy consumption. Furthermore, as the weighting factor for energy is incrementally increased, the reductions in optimal energy consumption diminish progressively, eventually converging. Figure 3 (b) depicts the average delay experienced under various weighting factors for delay. Consistently, the proposed approach yields

the minimal delay, surpassing the performance of the alternating optimization method. Notably, the strategy focusing solely on optimizing α demonstrates superior results compared to that which exclusively optimizes resource allocation. This advantage is attributed to the enhanced computational capabilities of the edge servers, which significantly reduce computational delays. Figure 3 (c) illustrates the mean stability of the model across a range of weighting factors for model stability. The method we proposed consistently attains the highest level of model stability. The alternating optimization approach outperforms the strategy that solely optimizes α a little, although both methods converge to nearly identical point in the long term. Conversely, the technique that focuses exclusively on optimizing resource allocation exhibits the poorest performance, primarily due to the arbitrary selection of the offloading decision α, which significantly impacts model stability.

5.3 The impact of the number of users and the number of edge servers

In this part, we assess the influence of both the user population and the number of edge servers on the effectiveness of the proposed approach to addressing the user-to-edge association challenge. We employ a comparative methodology as outlined below:

- **Baseline:** The baseline approach we choose is a greedy-based strategy. Under this strategy, each user opts for the edge server offering the highest available transmission rate, subject to bandwidth limitations.
- **Random:** The random user-to-edge server association method distributes users among edge servers in a stochastic manner, also adhering to bandwidth constraints.

In Figure 5 (a), we present the total energy consumption when there are different numbers of users. It can be observed that the proposed method always outperforms the two alternative strategies. The baseline strategy selects the edge server with the highest available transmission rate for each user; however, this approach may inadvertently overload servers that possess lower computational efficiency, thereby causing a marginal increase in energy consumption relative to our method. The random strategy invariably results in the highest energy expenditure. Subsequently, we analyze the average delay contingent on varying user quantities, as depicted in Figure 5 (b). It is evident that the proposed methodology consistently surpasses the two alternative strategies. Specifically, the random strategy yields the least favorable outcomes due to its arbitrary selection of edge servers for user allocation. While the baseline strategy may attain minimal communication delays, it tends to allocate an excessive number of users to a single edge server, thereby exacerbating the computational delays.

In Figure 4, the convergence performance of the algorithm is analyzed for the user-to-edge server association problem across varying quantities of edge servers. For each scenario considered, the user count remains constant at 100. The analysis reveals that the algorithm attains a stationary point as evidenced by the stabilization of the objective value. It is worth noticing that configurations with a smaller number of edge servers exhibit faster convergence rates. This enhanced speed of convergence can be attributed to the diminished complexity of the optimization challenge: a reduced number of servers correlates with fewer constraints and a lower number
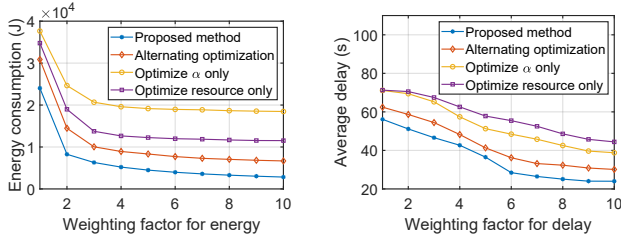
(a) The total energy consumption under different weighting factors for energy.

(b) The average delay under different weighting factors for energy.
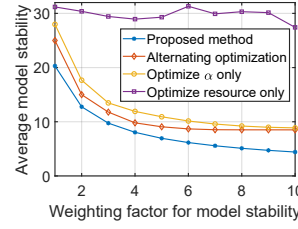
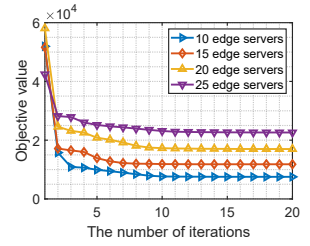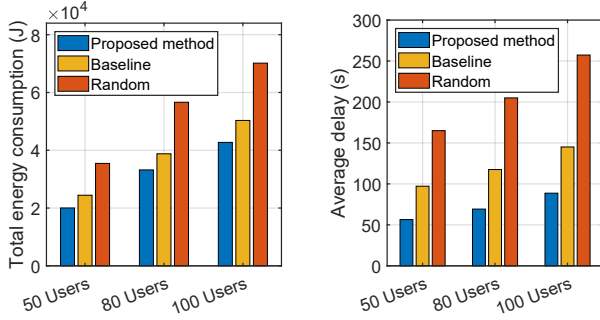(c) The average model stability under different weighting factors for model stability.

Figure 3: The performance of the proposed method under different weighting factors.

Figure 4: The convergence performance with different numbers of edge servers.

(a) The energy consumption under different numbers of users.

(b) The average delay under different numbers of users.

Figure 5: The performance of the algorithm under different numbers of mobile users.

of parameters requiring adjustment throughout the optimization procedure. In all tested configurations, the algorithm consistently achieved convergence within 13 iterations, thereby demonstrating its robust capability to efficiently resolve the user-to-edge server association problem.

6 CONCLUSION

In this study, we present a collaborative LLM training framework that merges the efforts of mobile users and edge servers. Here, mobile users are tasked with training the preliminary layers of the LLM, while the computationally intensive later layers are managed by the edge servers. We develop a multi-objective optimization strategy aimed at reducing overall energy usage and latency experienced by users, while also improving the stability of LLMs. Through analytical analysis, we establish an upper bound for the average-replace-one stability. The proposed algorithm leverages fractional programming and the CCCP method to derive solutions. Simulation results indicate that our approach effectively reduces energy usage and delay, and enhances the stability of LLMs in the mobile edge computing environments.

ACKNOWLEDGEMENT

A PROOF OF THEOREM 1

Denote $\mathbf{w}$ as the parameters of a language model, and $\xi = \dim(\mathbf{w})$ as the number of the parameters. When the users fine-tune a pretrained model, denoting $M^{\xi \times \xi}$ as a mask matrix which is a diagonal matrix with $M_{ii} = \{0, 1\}$ and $\mathcal{L}_S(\mathbf{w}) = \sum_{i=1}^{k} \ell(\mathbf{w}, z_i)$ for a training dataset S, the training process of fine-tuning the pre-trained model is to solve the following problem:

$$\min_{\mathbf{w}} \; \mathcal{L}_S(\mathbf{w}), \tag{A.1}$$

$$\text{s.t.} \; \|(I-M)(\mathbf{w} - \mathbf{w}^0)\|^2 = 0, \tag{A.1a}$$

where $\mathbf{w}^0$ are the parameters of the pre-trained model. According to the Lagrangian duality, problem (A.1) is equivalent to:

$$\min_{\mathbf{w}} \max_{\lambda} \; \mathcal{L}_S(\mathbf{w}) + \lambda \|(I - M)(\mathbf{w} - \mathbf{w}^0)\|^2, \tag{A.2}$$

where $\lambda \geq 0$ is the Lagrangian multiplier. Since

$$\min_{\mathbf{w}} \max_{\lambda} \; \mathcal{L}_S(\mathbf{w}) + \lambda \|(I - M)(\mathbf{w} - \mathbf{w}^0)\|^2 \geq$$
$$\min_{\mathbf{w}} \; \mathcal{L}_S(\mathbf{w}) + \|(I - M)(\mathbf{w} - \mathbf{w}^0)\|^2, \tag{A.3}$$

we then focus on optimizing the lower bound of problem (A.1), which is given by:

$$\min_{\mathbf{w}} \; \mathcal{L}'_S(\mathbf{w}) = \mathcal{L}_S(\mathbf{w}) + \|(I - M)(\mathbf{w} - \mathbf{w}^0)\|^2. \tag{A.4}$$

It indicates that minimizing initial loss function $\mathcal{L}_S(\mathbf{w})$, augmented by the regularization term $\|(I - M)(\mathbf{w} - \mathbf{w}^0)\|^2$ provides a lower bound on the optimal value of problem (A.1).

By taking the expectation with respect to M, we can get:

$$\mathbb{E}_M\left(\mathcal{L}'_S(\mathbf{w})\right) = \mathcal{L}_S(\mathbf{w}) + \mathbb{E}\|(I - M)(\mathbf{w} - \mathbf{w}^0)\|^2$$
$$= \mathcal{L}_S(\mathbf{w}) + \|(\mathbf{w} - \mathbf{w}^0)\|^2 \mathbb{E}\left(\sum_{i=1}^{\xi} (1 - M_{ii})^2\right)$$
$$= \mathcal{L}_S(\mathbf{w}) + \|(\mathbf{w} - \mathbf{w}^0)\|^2 (1 - \alpha), \tag{A.5}$$

where the validity of the last equality is attributed to the fact that the fraction of parameters subjected to fine-tuning is α. Therefore, $\mathcal{A}(S)$ can be given by:

$$\mathcal{A}(S) = \operatorname*{argmin}_{\mathbf{w}} \; \mathcal{L}_S(\mathbf{w}) + (1 - \alpha)\|\mathbf{w} - \mathbf{w}^0\|^2. \tag{A.6}$$

Next, we denote $f_S(\mathbf{w}) = \mathcal{L}_S(\mathbf{w}) + (1-\alpha)\|\mathbf{w} - \mathbf{w}^0\|^2$. Subsequently, $\forall \mathbf{u}, \mathbf{v}, \forall i \in \{1, \ldots, k\}$, we can get:

$$f_S(\mathbf{u}) - f_S(\mathbf{v})$$
$$= \mathcal{L}_S(\mathbf{u}) + (1-\alpha)\|\mathbf{u} - \mathbf{u}^0\|^2 - \left(\mathcal{L}_S(\mathbf{v}) + (1-\alpha)\|\mathbf{v} - \mathbf{v}^0\|^2\right)$$
$$= \mathcal{L}_{S^i}(\mathbf{u}) + (1-\alpha)\|\mathbf{u} - \mathbf{u}^0\|^2 - \left(\mathcal{L}_{S^i}(\mathbf{v}) + (1-\alpha)\|\mathbf{v} - \mathbf{v}^0\|^2\right)$$
$$- \frac{\ell(\mathbf{u}, z'_i) - \ell(\mathbf{v}, z'_i)}{k} + \frac{\ell(\mathbf{u}, z_i) - \ell(\mathbf{v}, z_i)}{k}$$
$$= f_{S^i}(\mathbf{u}) - f_{S^i}(\mathbf{v}) - \frac{\ell(\mathbf{u}, z'_i) - \ell(\mathbf{v}, z'_i)}{k} + \frac{\ell(\mathbf{u}, z_i) - \ell(\mathbf{v}, z_i)}{k}. \tag{A.7}$$

Let's set $\mathbf{u} = \mathcal{A}(\mathcal{S}^i)$ and $\mathbf{v} = \mathcal{A}(\mathcal{S})$ in (A.7). Using the fact that $f_{\mathcal{S}^i}(\mathbf{u}) \leq f_{\mathcal{S}^i}(\mathbf{v})$ because $\mathcal{A}(\mathcal{S}^i)$ is the minimizer of $f_{\mathcal{S}^i}(\mathbf{w})$, we can get:

$$f_{\mathcal{S}}(\mathcal{A}(\mathcal{S}^i)) - f_{\mathcal{S}}(\mathcal{A}(\mathcal{S})) \leq$$

$$\frac{\ell(\mathcal{A}(\mathcal{S}^i), z_i) - \ell(\mathcal{A}(\mathcal{S}), z_i)}{k} - \frac{\ell(\mathcal{A}(\mathcal{S}^i), z_i') - \ell(\mathcal{A}(\mathcal{S}), z_i')}{k}. \quad (A.8)$$

Since the loss function is strong convex, $f_{\mathcal{S}}$ is $2(1-\alpha)$-strong convex, which means $\forall \mathbf{u}, \mathbf{v}$,

$$f_{\mathcal{S}}(\mathbf{u}) \geq f_{\mathcal{S}}(\mathbf{v}) + \nabla f_{\mathcal{S}}(\mathbf{v})^\mathsf{T}(\mathbf{u} - \mathbf{v}) + (1-\alpha)\|\mathbf{u} - \mathbf{v}\|^2. \quad (A.9)$$

Let $\mathbf{u} = \mathcal{A}(\mathcal{S}^i)$ and $\mathbf{v} = \mathcal{A}(\mathcal{S})$, then $\nabla f_{\mathcal{S}}(\mathcal{A}(\mathcal{S})) = 0$ since $\mathcal{A}(\mathcal{S})$ is the minimizer of $f_{\mathcal{S}}(\mathbf{w})$. Therefore, (A.9) becomes:

$$f_{\mathcal{S}}(\mathcal{A}(\mathcal{S}^i)) \geq f_{\mathcal{S}}(\mathcal{A}(\mathcal{S})) + (1-\alpha)\|\mathcal{A}(\mathcal{S}^i) - \mathcal{A}(\mathcal{S})\|^2, \quad (A.10)$$

which can be rearranged as follows:

$$f_{\mathcal{S}}(\mathcal{A}(\mathcal{S}^i)) - f_{\mathcal{S}}(\mathcal{A}(\mathcal{S})) \geq (1-\alpha)\|\mathcal{A}(\mathcal{S}^i) - \mathcal{A}(\mathcal{S})\|^2. \quad (A.11)$$

Combing (A.8) and (A.11), it yields:

$$(1-\alpha)\|\mathcal{A}(\mathcal{S}^i) - \mathcal{A}(\mathcal{S})\|^2 \leq$$

$$\frac{\ell(\mathcal{A}(\mathcal{S}^i), z_i) - \ell(\mathcal{A}(\mathcal{S}), z_i)}{k} - \frac{\ell(\mathcal{A}(\mathcal{S}^i), z_i') - \ell(\mathcal{A}(\mathcal{S}), z_i')}{k}. \quad (A.12)$$

Since we assume $\ell(\cdot, z_i)$ is L-Lipschitz, which means:

$$\ell(\mathcal{A}(\mathcal{S}^i), z_i) - \ell(\mathcal{A}(\mathcal{S}), z_i) \leq L\|\mathcal{A}(\mathcal{S}^i) - \mathcal{A}(\mathcal{S})\|, \quad (A.13)$$

and

$$\ell(\mathcal{A}(\mathcal{S}), z_i') - \ell(\mathcal{A}(\mathcal{S}^i), z_i') \leq L\|\mathcal{A}(\mathcal{S}^i) - \mathcal{A}(\mathcal{S})\|. \quad (A.14)$$

Substituting (A.13) and (A.14) into (A.12) yields the following result:

$$(1-\alpha)\|\mathcal{A}(\mathcal{S}^i) - \mathcal{A}(\mathcal{S})\|^2 \leq \frac{2L\|\mathcal{A}(\mathcal{S}^i) - \mathcal{A}(\mathcal{S})\|}{k}, \quad (A.15)$$

which indicates:

$$\|\mathcal{A}(\mathcal{S}^i) - \mathcal{A}(\mathcal{S})\| \leq \frac{2L}{(1-\alpha)k}. \quad (A.16)$$

Inserting (A.16) into (A.13) leads us to determine the following:

$$\ell(\mathcal{A}(\mathcal{S}^i), z_i) - \ell(\mathcal{A}(\mathcal{S}), z_i) \leq \frac{2L^2}{(1-\alpha)k}. \quad (A.17)$$

Given that this is true for any $\mathcal{S}$ and z_i, we can finally deduce that $\forall i \in \{1, \ldots, k\}$:

$$\mathbb{E}_{\mathcal{S}}\big[|\ell(\mathcal{A}(\mathcal{S}), z_i) - \ell(\mathcal{A}(\mathcal{S}^i), z_i)|\big] \leq \frac{2L^2}{(1-\alpha)k}. \quad (A.18)$$

REFERENCES

[1] LT Hoai An, HV Ngai, and PD Tao. 2012. Exact penalty and error bounds in DC programming. *Journal of Global Optimization* 52, 3 (2012), 509–535.

[2] Dongqi Cai, Yaozong Wu, Shangguang Wang, Felix Xiaozhu Lin, and Mengwei Xu. 2023. Efficient federated learning for modern NLP. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–16.

[3] Min Chen and Yixue Hao. 2018. Task offloading for mobile edge computing in software defined ultra-dense network. *IEEE Journal on Selected Areas in Communications* 36, 3 (2018), 587–597.

[4] Thomas M Cover. 1999. *Elements of Information Theory*. John Wiley & Sons.

[5] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904* (2022).

[6] Thinh Quang Dinh, Jianhua Tang, Quang Duy La, and Tony QS Quek. 2017. Offloading in mobile edge computing: Task allocation and computational frequency scaling. *IEEE Transactions on Communications* 65, 8 (2017), 3571–3584.

[7] Zihao Fu, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. 2023. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 12799–12807.

[8] Michael Grant, Stephen Boyd, and Yinyu Ye. 2008. CVX: Matlab software for disciplined convex programming.

[9] Wenjuan Han, Bo Pang, and Ying Nian Wu. 2021. Robust transfer learning with pretrained language models through adapters. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. 854–861.

[10] Muhammad Fainan Hanif, Zhiguo Ding, Tharmalingam Ratnarajah, and George K Karagiannidis. 2015. A minorization-maximization method for optimizing sum rate in the downlink of non-orthogonal multiple access systems. *IEEE Transactions on Signal Processing* 64, 1 (2015), 76–88.

[11] Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. *arXiv preprint arXiv:2106.03164* (2021).

[12] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International conference on machine learning*. PMLR, 2790–2799.

[13] Yunpeng Huang, Yaonan Gu, Jingwei Xu, Zhihong Zhu, Zhaorun Chen, and Xiaoxing Ma. 2024. Securing reliability: A brief overview on enhancing in-context learning for foundation models. *arXiv preprint arXiv:2402.17671* (2024).

[14] Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2019. Mixout: Effective regularization to finetune large-scale pretrained language models. *arXiv preprint arXiv:1909.11299* (2019).

[15] Chang Liu and Jun Zhao. 2024. Resource allocation in large language model integrated 6G vehicular networks. In *IEEE 99th Vehicular Technology Conference (VTC2024-Spring)*. IEEE.

[16] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*.

[17] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602* (2021).

[18] Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, et al. 2024. MobileLLM: Optimizing sub-billion parameter language models for on-device use cases. In *Forty-first International Conference on Machine Learning*.

[19] Ruilong Ma, Jingyu Wang, Qi Qi, Xiang Yang, Haifeng Sun, Zirui Zhuang, and Jianxin Liao. 2023. Poster: PipeLLM: Pipeline LLM inference on heterogeneous devices with sequence slicing. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 1126–1128.

[20] Yuyi Mao, Jun Zhang, and Khaled B Letaief. 2016. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications* 34, 12 (2016), 3590–3605.

[21] Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines. *arXiv preprint arXiv:2006.04884* (2020).

[22] Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. 2023. Few-shot fine-tuning vs. in-context learning: A Fair comparison and evaluation. In *Findings of the Association for Computational Linguistics: ACL*.

[23] Matan Schliserman and Tomer Koren. 2022. Stability vs implicit bias of gradient methods on separable data and beyond. In *Proceedings of Thirty Fifth Conference on Learning Theory (Proceedings of Machine Learning Research, Vol. 178)*. PMLR, 3380–3394.

[24] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge university press.

[25] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. 2010. Learnability, stability and uniform convergence. *The Journal of Machine Learning Research* 11 (2010), 2635–2670.

[26] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge computing: Vision and challenges. *IEEE Internet of Things Journal* 3, 5 (2016), 637–646.

[27] Le Thi Hoai An and Pham Dinh Tao. 1997. Solving a class of linearly constrained indefinite quadratic problems by DC algorithms. *Journal of global optimization* 11 (1997), 253–285.

[28] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[29] Feng Wang, Jie Xu, Xin Wang, and Shuguang Cui. 2017. Joint offloading and computing optimization in wireless powered mobile-edge computing systems. *IEEE Transactions on Wireless Communications* 17, 3 (2017), 1784–1797.

[30] Daliang Xu, Wangsong Yin, Xin Jin, Ying Zhang, Shiyun Wei, Mengwei Xu, and Xuanzhe Liu. 2023. LLMCad: Fast and scalable on-device large language model inference. *arXiv preprint arXiv:2309.04255* (2023).

[31] Xincao Xu, Kai Liu, Penglin Dai, Feiyu Jin, Hualing Ren, Choujun Zhan, and Songtao Guo. 2023. Joint task offloading and resource optimization in NOMA-based vehicular edge computing: A game-theoretic DRL approach. *Journal of Systems Architecture* 134 (2023), 102780.

[32] Cheng Zhan, Han Hu, Xiufeng Sui, Zhi Liu, and Dusit Niyato. 2020. Completion time and energy optimization in the UAV-enabled mobile-edge computing system. *IEEE Internet of Things Journal* 7, 8 (2020), 7808–7822.

[33] Jun Zhao, Liangxin Qian, and Wenhan Yu. 2024. Human-centric resource allocation in the Metaverse over wireless communications. *IEEE Journal on Selected Areas in Communications (JSAC)* 42, 3 (2024), 514–537. https://arxiv.org/pdf/2304.00355.pdf

[34] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*. PMLR, 12697–12706.