Novel Transform Based Optimization for Resource Allocation and Task Offloading in Communication Networks

Yitong Wang and Jun Zhao, *Member, IEEE*, Liangxin Qian, and Chang Liu

***Abstract*—In wireless communication and edge computing networks, fractional programming (FP) and multiplicative programming (MP) are fundamental methodologies widely employed in solving non-convex optimization problems. Prior work introduced a remarkable method to solve non-convex functions with multiple ratios by deriving novel tight upper bounds, which we formalize as the `UpperBound` transform in this paper. However, the `UpperBound` transform faces critical limitations, particularly when directly extended to MP problems involving non-negative functions or discrete optimization variables. In this paper, we introduce a generalized transform termed the `UP` transform to overcome these limitations. We rigorously prove that the `UP` transform guarantees convergence to a Karush-Kuhn-Tucker (KKT) point for a broader class of MP problems, including scenarios where variables can be zero or discrete. We comprehensively illustrate the `UP` transform's utility through two practical applications: partial task offloading in mobile edge computing, optimizing computation and energy efficiency; and user association coupled with resource allocation in heterogeneous networks, addressing mixed discrete-continuous optimization challenges. Comparative evaluations against conventional methods demonstrate superior convergence speed, efficiency, solution quality, and reduced computational complexity of the proposed `UP` transform based algorithms.**

***Index Terms*—Fractional programming, multiplicative programming, non-convex optimization, wireless networks, edge computing.**

I. Introduction

FRACTIONAL programming (FP), which solves the optimization problem whose objective and constraint functions consist of ratio term(s), is a critical tool in developing current wireless networks and enhancing the performance of communication services [1]. In practical communication and networking systems, FP is particularly suitable for ratio-based optimization objectives, such as maximizing resource utilization efficiency (e.g., utility-to-cost ratio, throughput-to-power ratio). General FP optimization problem is formulated as follows with $A_n(\boldsymbol{x}) \geq 0$, $B_n(\boldsymbol{x}) > 0$, and $G(\boldsymbol{x})$ being scalar functions of a variable $x \in \mathbb{R}^M$, for $n = 1, 2, \ldots, N$, where N is a positive integer:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & K(\boldsymbol{x}) \stackrel{\text{def}}{=} G(\boldsymbol{x}) + \sum_{n=1}^{N} \frac{A_n(\boldsymbol{x})}{B_n(\boldsymbol{x})} \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{X}, \end{aligned} \tag{1}$$

where $\mathcal{X} \subseteq \mathbb{R}^M$ is a closed feasible set. The objective function in (1) is typically non-convex of the optimization variable $\boldsymbol{x}$ due to the sum of the N ratios in (1), as explained in [2], [3]. To derive effective resource allocation schemes of communication networks, constructing tractable surrogate functions to tightly approximate the non-convex objective functions of the problem (1) is a conventional idea for finding the optimal solutions [4].

Yitong Wang, Jun Zhao, Liangxin Qian, and Chang Liu are with Nanyang Technological University, Singapore. Email: yitong002@e.ntu.edu.sg, junzhaocmu@163.com, qian0080@e.ntu.edu.sg, liuc0063@e.ntu.edu.sg

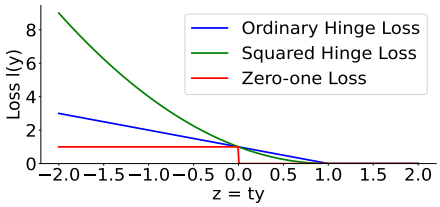


Fig. 1: Hinge loss and zero-one loss functions: zero-one loss function for fixed $t = 1$ is red. Two variants of the hinge loss as a function of $z = ty$: the ordinary variant is blue, and its square is green.

The idea of finding surrogate functions as the upper bound of non-convex functions also appears in machine learning. In Support Vector Machines (SVM), the hinge loss acts as a continuous upper bound to the zero-one loss, designed to maximize classification margins. Its ordinary variant is defined as $\max(0, 1 - t \cdot y)$, where t represents the ground truth label and y is the predicted value, respectively. The key approach in optimizing the zero-one loss involves indirectly controlling the actual loss by optimizing its hinge loss. Fig. 1 presents zero-one loss functions and two variants of hinge loss functions. Since the hinge loss is non-negative and convex, its expectation serves as an upper bound on the actual loss; therefore, minimizing this surrogate effectively controls the overall loss. Thus, this idea in machine learning can extend to non-convex optimization techniques, ensuring both computational efficiency and convergence to an optimal solution.

Zhao *et al.* [2] in IEEE JSAC 2024 enhanced the performance of the human-centric Metaverse by minimizing the energy consumption and maximizing the utility-cost ratio (UCR) of the system. The proposed algorithm was inspired by a similar idea of choosing a well-approximated upper bound of the non-convex functions to ensure convergence of the algorithm. However, this transform is highly dependent on the specific settings and has not been abstracted into a theorem that can guide and be adopted into a series of FP optimization problems formulated on networking. In Theorem 1 of Section II-A, we give the details of the abstracted theorem and refer to this transform as the `UpperBound` transform.

Multiplicative programming (MP), as another foundational tool, is more appropriate for problems involving product terms between decision variables, such as the joint optimization of task offloading ratios and communication/computation resources. Such product terms typically arise in edge computing, joint scheduling, or load balancing scenarios where perfor-

mance metrics depend on the product of multiple factors. To further unlock the potential of UpperBound transform, we consider the MP optimization problems, which can typically be formulated as

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & H(\boldsymbol{x}) \overset{\text{def}}{=} G(\boldsymbol{x}) + \sum_{n=1}^{N} [A_n(\boldsymbol{x}) B_n(\boldsymbol{x})] \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{X}, \end{aligned}$$

where the objective function is also typically non-convex with respect to the optimization variable $\boldsymbol{x}$.

However, when adopting the idea of finding the upper bound of the non-convex functions in MP problems [5], some inherent characters existing in FP problems are destroyed, resulting in the proposed algorithm of MP problems with UpperBound transform failing to converge. We illustrate the inherent characters and corresponding limitations in Section II and update the UpperBound transform to make it more general and adaptable to more general scenarios. Updated UpperBound transform is referred to as UP transform, and its details are presented in Theorem 2.

Additionally, recent works [6]–[8] on resource allocation in wireless communication networks also focus on solving non-convex optimization problems. Specifically, these studies generally adopt the block coordinate descent (BCD) method and propose an iterative algorithm where a single block of the variables (consisting of objective variables and/or introduced auxiliary variables) is optimized while the remaining blocks are kept fixed during each iteration. Different updating orders of blocks can significantly affect the convergence rate of the algorithm. In particular, when different blocks are strongly coupled, an inappropriate updating order may result in ill-conditioned subproblems or overly conservative surrogate functions, leading to zig-zagging trajectories in the optimization path [9]. This oscillatory behavior typically results in inefficient descent directions and significantly slows down convergence. Moreover, in surrogate-based methods involving auxiliary variables, updating certain blocks based on outdated surrogate information can further aggravate this effect, causing large disparities in convergence speed across different updating schedules. These observations motivate the need for optimization frameworks that reduce sensitivity to block updating order and provide more stable and efficient convergence. The important **novelty** of our proposed UP transform is that all the proposed optimization variables can be optimized in one single block at each iteration of the proposed BCD algorithm, and the rest of the introduced variables (i.e., auxiliary variables) are optimized in another block.

For **goals** of this paper, we first abstract the UpperBound transform and introduce the algorithm of adopting this transform. Then, we illustrate the challenges when directly adopting UpperBound transform to MP optimization problems and generalize it to the UP transform. We furthermore prove the effectiveness of our proposed transforms in typical communication networks. The main **contributions** are as follows:

- We abstract the UpperBound transform and introduce the adoption of this transform in MP problems with the same settings of functions $A_n(\boldsymbol{x})$ and $B_n(\boldsymbol{x})$ as our recent work [2]. We then illustrate the details of the optimization algorithm and clarify the correspondence between the primal optimization problem and the recast problem.
- We refine the UpperBound transform to UP transform and focus on the complicated MP optimization problems of $B_n(\boldsymbol{x}) = 0$ with discrete or mixed discrete-continuous variables. After illustrating and resolving corresponding challenges, we analyze the essential properties of UP transform, verify it converges to a KKT point, and further present the extended results of this transform.
- To validate effectiveness, we apply the UP transform to the minimization problems of a latency-sensitive network, jointly optimizing partial offloading rates and computation frequencies (i.e., the product of the offloading ratio and communication cost). We obtain the optimum by adopting KKT conditions analysis and then analyze the computational complexity of the UP transform-based algorithm.
- We apply the UP transform to a heterogeneous network (HetNet) to address non-convex problems with mixed discrete-continuous variables. Specifically, we minimize energy consumption and processing delay by jointly optimizing user association, offloading decisions, computation resources, and transmission power. To tackle this NP-hard problem, we propose a novel algorithm integrating the UP transform with successive convex approximation (SCA).

Notation. Let $\mathbb{R}$ be the set of real numbers, while $\mathbb{R}_+$ (resp., $\mathbb{R}_{++}$) denotes the set of non-negative (resp., positive) real numbers; i.e., $\mathbb{R}_+ = [0, \infty)$ and $\mathbb{R}_{++} = (0, \infty)$. We use boldfaced lower-case letters (e.g., $\boldsymbol{x}$) to denote vectors. For an M-dimensional vector $\boldsymbol{x}$, its m-th dimension is given by x_m. The symbol "$\overset{\text{def}}{=}$" and ":=" mean "by definition".

Roadmap. The rest of the paper is organized as follows: Section II presents the intuitions and main results of our proposed transforms. Section III introduces the guidelines for adopting proposed transforms in different practical scenarios of wireless communication networks, and further evaluates and presents the effectiveness of our proposed transforms. Section IV compares the computational complexity of the proposed algorithm with traditional optimization techniques. Improvements compared with prior related papers are given in Section V. Section VI concludes this paper.

II. Main Results

In this section, we first give an intuitive overview of the proposed transforms introduced in Theorems 1 and 2, and then present the details of these theorems.

Intuition of Theorem 1 (UpperBound transform). In Theorem 1, we propose an approach to solve a non-convex optimization problem which involves multiplicative term(s)[1,2]. Such a problem belongs to the class of **MP problems**. Specifically, in an MP problem $\mathbb{P}_1$ with the variable being a vector $\boldsymbol{x}$, we can replace any multiplicative term $A_n(\boldsymbol{x})B_n(\boldsymbol{x})$ ("n" is the index for each multiplicative term) by approximation functions

$$K_n(\boldsymbol{x}, y_n) \overset{\text{def}}{=} [A_n(\boldsymbol{x})]^2 y_n + \frac{[B_n(\boldsymbol{x})]^2}{4y_n}, \tag{2}$$

[1]Multiplicative term(s) are typically non-convex. For instance, even the simple product $f(x, y) \overset{\text{def}}{=} x \times y$ is neither jointly convex nor concave with respect to x and y, because $f(2,2) < \frac{f(1,1)+f(3,3)}{2}$ and $f(2,2) > \frac{f(1,3)+f(3,1)}{2}$.

[2]Section II-D1 shows that fraction term(s) can also be handled by our optimization approach.

where $\boldsymbol{x} \in \mathcal{X}, y_n \in \mathbb{R}_{++}$ and solve the resulting problem $\mathbb{P}_2$ with the variables being $\boldsymbol{x}$ and $\boldsymbol{y}$, for $\boldsymbol{y} \overset{\text{def}}{=} [y_1, y_2, \ldots, y_N]$. Here, y_n serves as an auxiliary variable introduced to construct the convex surrogate function $K_n(x, y_n)$. It acts as a parameter that decouples the multiplicative terms, allowing the non-convex objective to be tractably solved.

Intuition of Theorem 2 (UP transform). Theorem 1 presents a way to handle the non-convex multiplicative term(s) in MP problems, but requires both the multiplier and multiplicand to be strictly positive. As will be explained at the beginning of Section II-C, when one of the multiplier and multiplicand takes the zero value, Theorem 1 does not work, and this motivates us to propose Theorem 2 as a refinement.

Applicability to general optimization involving multiplicative or fractional terms. We present Theorems 1 and 2 in a way of solving MP. Actually, as we will discuss in Section II-D, our approaches in Theorems 1 and 2 can be used for solving **fractional programming (FP)** [10], where FP is related to optimize a ratio, sum of ratios, or related functions. In fact, our approaches are generally applicable to any optimization problems involving multiplicative or fractional terms, not limited to common MP or FP problems.

A. Theorem 1 (UpperBound Transform)

Let $\mathcal{N}$ denote the set $\{1, 2, ..., N\}$. We now present Theorem 1, and some parts are placed in boxes since we will replace them when we present Theorem 2 in Section II-C. In this way, we reuse other parts of Theorem 1 when stating Theorem 2 without repeating those texts.

Theorem 1. *Let $\boldsymbol{x} \in \mathbb{R}^M$ be an M-dimensional real vector. Functions $A_n(\boldsymbol{x}) : \mathbb{R}^M \to \mathbb{R}_{++}$, $\boxed{B_n(\boldsymbol{x}) : \mathbb{R}^M \to \mathbb{R}_{++}}$ for $n \in \mathcal{N}$, and $G(\boldsymbol{x}) : \mathbb{R}^M \to \mathbb{R}$ all have definitions on a feasible set $\mathcal{X} \subseteq \mathbb{R}^M$. We aim to solve the following minimization problem $\mathbb{P}_1$:*

$$\text{Problem } \mathbb{P}_1\text{: } \underset{\boldsymbol{x}}{\text{minimize}}\ H(\boldsymbol{x}) \text{ subject to } \boldsymbol{x} \in \mathcal{X}, \tag{3}$$

$$\text{for } H(\boldsymbol{x}) \overset{def}{=} G(\boldsymbol{x}) + \sum_{n=1}^{N} [A_n(\boldsymbol{x}) B_n(\boldsymbol{x})] \tag{4}$$

To solve Problem $\mathbb{P}_1$, we turn to reformulated Problem $\mathbb{P}_2$ below. After defining $K_n(\boldsymbol{x}, y_n)$ based on (2) for $n \in \mathcal{N}$, and further defining

$$W(\boldsymbol{x}, \boldsymbol{y}) \overset{def}{=} G(\boldsymbol{x}) + \sum_{n=1}^{N} K_n(\boldsymbol{x}, y_n), \tag{5}$$

where $\boldsymbol{y} \overset{def}{=} [y_1, y_2, \ldots, y_N]$, we consider

$$\text{Problem } \mathbb{P}_2\text{: } \underset{\boldsymbol{x}, \boldsymbol{y}}{\text{minimize}}\ W(\boldsymbol{x}, \boldsymbol{y}) \tag{6}$$

$$\text{subject to } \boldsymbol{x} \in \mathcal{X} \text{ and } \boldsymbol{y} \in \mathbb{R}^N_{++}. \tag{6a}$$

We solve $\mathbb{P}_2$ using an alternating optimization (AO) process: With an initial $\boldsymbol{x}^{(0)} \in \mathcal{X}$, we perform the following operations iteratively to get $\boldsymbol{y}^{(j)}$ and $\boldsymbol{x}^{(j)}$ for $j = 1, 2, \ldots$:

① *Fix $\boldsymbol{x}^{(j-1)}$ and solve Problem $\mathbb{P}_2(\boldsymbol{x}^{(j-1)})$ to obtain $\boldsymbol{y}^{(j)} \overset{def}{=} \left[y_1^{(j)}, \ldots, y_N^{(j)}\right]$ for*

$$y_n^{(j)} \overset{def}{=} \frac{B_n(\boldsymbol{x}^{(j-1)})}{2A_n(\boldsymbol{x}^{(j-1)})},\ \forall n \in \mathcal{N}, \tag{7}$$

which is the closed-form minimizer of $W(\boldsymbol{x}^{(j-1)}, \boldsymbol{y})$ with respect to $\boldsymbol{y}$.

It is clear that $\boldsymbol{y}^{(j)}$ optimally solves $\mathbb{P}_2(\boldsymbol{x}^{(j-1)})$, with $\mathbb{P}_2(\boldsymbol{x})$ denoting optimizing $\boldsymbol{y}$ for $\mathbb{P}_2$ given $\boldsymbol{x}$; i.e.,

$$\text{Problem } \mathbb{P}_2(\boldsymbol{x})\text{: } \underset{\boldsymbol{y}}{\text{minimize}}\ W(\boldsymbol{x}, \boldsymbol{y}) \text{ subject to } \boldsymbol{y} \in \mathbb{R}^N_{++}; \tag{8}$$

② *Fix $\boldsymbol{y}^{(j)}$ and solve Problem $\mathbb{P}_2(\boldsymbol{y}^{(j)})$ to obtain $\boldsymbol{x}^{(j)}$ as a KKT point of $\mathbb{P}_2(\boldsymbol{y}^{(j)})$, where $\mathbb{P}_2(\boldsymbol{y})$ means optimizing $\boldsymbol{x}$ for $\mathbb{P}_2$ given $\boldsymbol{y}$; i.e.,*

$$\text{Problem } \mathbb{P}_2(\boldsymbol{y})\text{: } \underset{\boldsymbol{x}}{\text{minimize}}\ W(\boldsymbol{x}, \boldsymbol{y}) \text{ subject to } \boldsymbol{x} \in \mathcal{X}. \tag{9}$$

Then we have the following result on the relationship between Problems $\mathbb{P}_1$ and $\mathbb{P}_2$:

Result:
With $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ denoting the convergence[a] of $(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})$ as $j \to \infty$, then $\boldsymbol{x}^*$ is a KKT point for Problem $\mathbb{P}_1$. Hence, supposing the convergence criteria[b] is met after J iterations (i.e., after running $j = 1, 2, \ldots, J$), we can consider $\boldsymbol{x}^{(J)}$ as solving $\mathbb{P}_1$ by finding a KKT point (subject to arbitrarily small error tolerance).

[a] From "①" and "②" above, $(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})|_{j=1,2,\ldots}$ are from solving Problem $\mathbb{P}_2$ of (6) via alternating optimization (AO), and many studies on AO show the convergence of $(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})$ as $j \to \infty$; viz., the convergence of AO for $(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})$ is guaranteed [11].

[b] An example convergence criteria is that objective-function values under $(\boldsymbol{x}^{(J-1)}, \boldsymbol{y}^{(J-1)})$ and $(\boldsymbol{x}^{(J)}, \boldsymbol{y}^{(J)})$ respectively differ negligibly; e.g., when $W(\boldsymbol{x}^{(J)}, \boldsymbol{y}^{(J)})/W(\boldsymbol{x}^{(J-1)}, \boldsymbol{y}^{(J-1)})$ is close to 1.

Proof: Please refer to Appendix A. □

Remark 1. *$A_n(\boldsymbol{x})B_n(\boldsymbol{x})$ also covers $A_n(\boldsymbol{y})B_n(\boldsymbol{z})$. When considering $A_n(\boldsymbol{y})B_n(\boldsymbol{z})$, the process of forming $\boldsymbol{x}$ is by concatenating variables $\boldsymbol{y}$ and $\boldsymbol{z}$.*

Remark 2. *UpperBound transform based on approximation (2) keeps tight since for each $A_n(\boldsymbol{x})B_n(\boldsymbol{x})$, surrogate function $K_n(\boldsymbol{x}, y_n)$ serves as a valid upper bound with equality attained at $y_n^* = \frac{B_n(\boldsymbol{x})}{2A_n(\boldsymbol{x})}$. For any fixed $\boldsymbol{x}$, each multiplicative term admits the exact representation $A_n(\boldsymbol{x})B_n(\boldsymbol{x}) = \min_{y_n > 0} K_n(\boldsymbol{x}, y_n)$, which establishes the tightness of the transform. Moreover, during AO process, operation ② satisfies the standard majorization–minimization (MM) conditions, rendering the surrogate tight at each iterate.*

Remark 3. *It is worth clarifying that Problem $\mathbb{P}_2$ constitutes not only an approximation but also an equivalent reformulation of Problem $\mathbb{P}_1$. Specifically, for any fixed $\boldsymbol{x}$, the minimization of the auxiliary function $K_n(\boldsymbol{x}, y_n)$ with respect to y_n admits a unique solution $y_n^* = B_n(\boldsymbol{x})/[2A_n(\boldsymbol{x})]$ (as derived in Eq. (7)). Substituting y_n^* back into Eq. (2) yields the exact identity: $\min_{y_n} K_n(\boldsymbol{x}, y_n) = [A_n(\boldsymbol{x})]^2 \frac{B_n(\boldsymbol{x})}{2A_n(\boldsymbol{x})} + \frac{[B_n(\boldsymbol{x})]^2}{4\frac{B_n(\boldsymbol{x})}{2A_n(\boldsymbol{x})}} = A_n(\boldsymbol{x})B_n(\boldsymbol{x})$. Consequently, minimizing $W(\boldsymbol{x}, \boldsymbol{y})$ over $\boldsymbol{y}$ recovers the original objective $H(\boldsymbol{x})$ exactly. Thus, the alternating optimization of $(\boldsymbol{x}, \boldsymbol{y})$ in $\mathbb{P}_2$ is mathematically equivalent to solving $\mathbb{P}_1$.*

B. Limitations of Theorem 1

Theorem 1 demonstrates the validity of the abstracted UpperBound transform under specific assumptions; however, applying this transform to practical problems arising in communication networks presents two key challenges:

- **Challenge 1 (Non-strict Positivity):** In network optimization problems, functions $A_n(\boldsymbol{x})$ or $B_n(\boldsymbol{x})$ typically is non-negative rather than strictly positive; that is, A_n, B_n :

$\mathbb{R}^M \to \mathbb{R}_+$ with $A_n(\boldsymbol{x}) = 0$ or $B_n(\boldsymbol{x}) = 0$ occurring at feasible points. In practical scenarios, this condition corresponds to many common situations (e.g., inactive links, null offloading decisions, or user disconnection). This highlights a fundamental limitation of the `UpperBound` transform: it cannot handle such cases. In such cases, the update process in AO may prevent convergence, and **numerical instability** may become a critical consideration during the iterations, undermining the correctness and stability of the transform:

  - **Division by Small Values:** When $A_n(\boldsymbol{x}) : \mathbb{R}^M \to \mathbb{R}_+$ holds, it is possible that $A_n(\boldsymbol{x}^{(j-1)})$ approaches zero during j-th iteration. In this case, the denominator of $y_n^{(j)}$ in the update step "①" becomes exceedingly small, causing $y_n^{(j)}$ in (7) to grow unboundedly large, breaking the convergence of the AO process.
  - **Zero Value Numerator:** When $B_n(\boldsymbol{x}) : \mathbb{R}^M \to \mathbb{R}_+$ holds, it is possible that $B_n(\boldsymbol{x}^{(j-1)}) = 0$ during update process. In this case, the update rule (7) yields $y_n^{(j)} = 0$, which leads to an undefined value of $K_n(\boldsymbol{x}, y_n^{(j)})$ due to a division by zero in its second term $\frac{[B_n(\boldsymbol{x})]^2}{4y_n}$. As a result, the surrogate objective $W(\boldsymbol{x}, \boldsymbol{y}^{(j)})$ becomes invalid, the subproblem $\mathbb{P}_2(\boldsymbol{y}^{(j)})$ is no longer well-defined, and the update step "②" can no longer be proceed..

- **Challenge 2 (Discrete or Mixed Domains):** Communication network problems often involve integer variables (e.g., user association indicators [6], [7]) or mixed discrete-continuous decision spaces. When $\boldsymbol{x}$ includes such discrete elements, the $\boldsymbol{x}$-subproblem in (9) in each AO step becomes a combinatorial optimization, potentially NP-hard, and no longer guarantees tractability or convergence under the framework of Theorem 1.

To address the limitations and Challenge 1 of the `UpperBound` transform, we propose a refined and generalized reformulation termed the `UP` transform, which is detailed in Section II-C. The resolution to Challenge 2 is provided in Section III-B. Before presenting this new theoretical framework, we first formalize the essential **properties** that the `UP` transform must satisfy to remain valid in general MP settings:

- Decoupling: The proposed `UP` transform should have the same decoupling property with the form $K_n(\boldsymbol{x}, y_n) = K_{n,1}(A_n(\boldsymbol{x}))f_1(y_n) + K_{n,2}(B_n(\boldsymbol{x}))f_2(y_n)$, where y_n is an auxiliary variable;
- Equivalent Solution: Variable $\boldsymbol{x}^*$ is the optimum of $A_n(\boldsymbol{x})B_n(\boldsymbol{x})$ if and only if $\boldsymbol{x}^*$ and y_n^* minimize $K_n(\boldsymbol{x}, y_n)$;
- Relaxed Objective: For $B_n(\boldsymbol{x}) > 0$ (resp. $B_n(\boldsymbol{x}) = 0$), $K_n(\boldsymbol{x}, y_n^*) = A_n(\boldsymbol{x})B_n(\boldsymbol{x})$ (resp. $\min_y K_n(\boldsymbol{x}, y_n)$) for variable $\boldsymbol{x}$ if and only if $y_n^* = \arg\min_{y_n} K_n(\boldsymbol{x}, y_n)$;
- Convexity: Function $K_n(\boldsymbol{x}, y_n)$ is convex of y_n for fixed variable $\boldsymbol{x}$, i.e., its *Hessian* matrix is positive semi-definite;
- Robustness: Updated transform is not only applicable for positive-value functions but also for non-negative functions, i.e., $\{\boldsymbol{x} \mid A_n(\boldsymbol{x}) \geq 0 \vee B_n(\boldsymbol{x}) \geq 0\} \subseteq \mathcal{X}$.

The above properties guarantee that the recast optimization problem after adopting `UP` transform converges to the same optimal solutions as the original optimization problem.

C. Theorem 2 (`UP` Transform)

We investigate Theorem 2 below, where the theoretical intuition and the core idea behind refining the `UpperBound` transform have already been discussed on Page 3 (the intuition subsection) and Section II-B, respectively.

From the perspective of practical wireless networks, the motivation for investigating `UP` transform becomes particularly pronounced. In many communication and edge-computing optimization problems, zero-valued terms naturally arise due to inactive links [12], null offloading decisions [13], user disconnections [14], or binary/mixed-integer decisions. The factors in a multiplicative term $A_n(\boldsymbol{x})B_n(\boldsymbol{x})$ are typically non-negative rather than strictly positive.

With this motivation, we consider the setting where either the multiplier $A_n(\boldsymbol{x})$ or the multiplicand $B_n(\boldsymbol{x})$ in $A_n(\boldsymbol{x})B_n(\boldsymbol{x})$ takes the zero value. Without loss of generality, we focus on the case $A_n(\boldsymbol{x}) > 0$ and $B_n(\boldsymbol{x}) \geq 0$. This does not reduce generality, since the complementary case $A_n(\boldsymbol{x}) \geq 0$ and $B_n(\boldsymbol{x}) > 0$ can be handled by swapping the two functions: define $A'_n(\boldsymbol{x}) \triangleq B_n(\boldsymbol{x}) > 0$ and $B'_n(\boldsymbol{x}) \triangleq A_n(\boldsymbol{x}) \geq 0$, and then analyze the equivalent product term $A'_n(\boldsymbol{x})B'_n(\boldsymbol{x})$ within the same framework.

Theorem 2. *First, we use "$\boxed{B_n(\boldsymbol{x}) : \mathbb{R}^M \to \mathbb{R}_+}$" to replace the boxed text "$\boxed{B_n(\boldsymbol{x}) : \mathbb{R}^M \to \mathbb{R}_{++}}$" of Theorem 1, but keep $A_n(\boldsymbol{x}) : \mathbb{R}^M \to \mathbb{R}_{++}$ of Theorem 1. Then with an initial $\boldsymbol{x}^{(0)} \in \mathcal{X}$, we iteratively perform the following operations to obtain $\boldsymbol{y}^{(j)}$ and $\boldsymbol{x}^{(j)}$ for $j = 1, 2, \ldots$:*

- *replace the boxed "①" of Theorem 1 by the boxed "❶" below to ensure that $y_n^{(j)}$ is strictly positive:*

> ❶ *Fix $\boldsymbol{x}^{(j-1)}$ and obtain $\boldsymbol{y}^{(j)} \overset{def}{=} [y_1^{(j)}, \ldots, y_N^{(j)}]$ with*
> $$y_n^{(j)} \overset{def}{=} \max\left\{\frac{B_n(\boldsymbol{x}^{(j-1)})}{2A_n(\boldsymbol{x}^{(j-1)})},\ c_n\right\},\ \forall n \in \mathcal{N}, \quad (10)$$
> *where $c_n > 0$ for $n \in \mathcal{N}$ are hyperparameters that we can adaptively choose;*

- *we still follow "②" of Theorem 1 to obtain $\boldsymbol{x}^{(j)}$ as a KKT point of $\mathbb{P}_2(\boldsymbol{y}^{(j)})$, for $\mathbb{P}_2(\boldsymbol{y})$ defined in (9).*

Finally, we can replace the boxed result part of Theorem 1 with the following:

> ***Results:***
> *(i) The sequence $(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})$ converges as $j \to \infty$;*
> *(ii) With $(\boldsymbol{x}^{\#}, \boldsymbol{y}^{\#})$ denoting the convergence of $(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})$ as $j \to \infty$, then $\boldsymbol{x}^{\#}$ is a KKT point for Problem $\mathbb{P}'_1$ defined as follows.*
> $$\mathbb{P}'_1\text{: } \underset{\boldsymbol{x}}{\text{minimize}}\ L_{\boldsymbol{c}}(\boldsymbol{x}) \text{ subject to } \boldsymbol{x} \in \mathcal{X}, \quad (11)$$
> $$\text{for } L_{\boldsymbol{c}}(\boldsymbol{x}) \overset{def}{=} G(\boldsymbol{x}) + \sum_{n=1}^{N} \max\left\{A_n(\boldsymbol{x})B_n(\boldsymbol{x}),\ [A_n(\boldsymbol{x})]^2 c_n + \frac{[B_n(\boldsymbol{x})]^2}{4c_n}\right\}, \quad (12)$$
> *where the subscript "$\boldsymbol{c}$" in $L_{\boldsymbol{c}}(\boldsymbol{x})$ means $[c_1, \ldots, c_N]$.*

(iii) $L_{\boldsymbol{c}}(\boldsymbol{x})$ is an upper bound of $H(\boldsymbol{x})$ in the sense that $L_{\boldsymbol{c}}(\boldsymbol{x}) \geq H(\boldsymbol{x})$ and $\lim_{\boldsymbol{c}\to\boldsymbol{0}} L_{\boldsymbol{c}}(\boldsymbol{x}) = H(\boldsymbol{x})$. Following the method of minimizing an upper bound to solve an optimization problem, which is widely used in various optimization and machine learning problems (viz., Section I), we consider the iterative process of updating $\boldsymbol{y}^{(j)}$ and $\boldsymbol{x}^{(j)}$ as a way to solve $\mathbb{P}_1$; i.e., $\boldsymbol{x}^{\#}$ in Result "(ii)" above can be considered a "decent" solution with appropriately set $\boldsymbol{c}$.

Proof: Please refer to Appendix B. □

Remark 4. *Result "(iii)" of Theorem 2 extends the theoretical validity to the boundary case where $B_n(\boldsymbol{x}) \geq 0$ while Theorem 1 establishes a KKT solution strictly under the condition of positive denominators ($A_n(\boldsymbol{x}) > 0, B_n(\boldsymbol{x}) > 0$). This relaxation is analytically significant as standard FP methods typically become undefined when the denominator equals zero. In practical cases, this relaxation encompasses scenarios involving binary variables (i.e., $B_n(\boldsymbol{x}) \in \{0, 1\}$), which are ubiquitous in resource allocation problems such as heterogeneous mobile edge computing. Consequently, the proposed framework is robust against singular cases. Leveraging this property, Section III develops an SCA-based algorithm that ensures convergence to a stationary point for the formulated discrete-continuous optimization problem.*

Remark 5. *The hyperparameters $c_n > 0$ for $n \in \mathcal{N}$ in (10) ensure that the surrogate function remains well-defined when $B_n(\boldsymbol{x}) = 0$ or $B_n(\boldsymbol{x})$ approaches zero. Specifically, the operator $\max(\cdot, c_n)$ prevents division by zero and unbounded auxiliary variables during the alternating optimization process. When the standard update value $\frac{B_n(\boldsymbol{x}^{(j-1)})}{2A_n(\boldsymbol{x}^{(j-1)})}$ is at or above c_n, $y_n^{(j)}$ preserves the exact tightness of the surrogate upper bound. Conversely, when the value falls below c_n, the transformation slightly sacrifices this tightness but significantly enhances numerical stability and robustness, preventing algorithmic divergence in cases of numerical instability.*

Remark 6. *The determination of c_n involves a trade-off between approximation tightness and numerical stability. Theoretically, c_n serves as a numerical safeguard. It must be strictly positive to prevent division-by-zero errors when the denominator approaches zero, yet sufficiently small to maintain the tightness of the surrogate upper bound (as established in Theorem 2, Result (iii)). Specifically, we set $c_n = \varepsilon, \forall n$, where ε is a small positive constant (e.g., on the order of 10^{-6}). This choice ensures that the auxiliary variable $y_n^{(j)}$ remains bounded while preserving the condition $y_n^{(j)} = \frac{B_n(x)}{2A_n(x)}$ for the vast majority of iterations where the links are active. Empirically, as demonstrated in the sensitivity analysis in Section III-E and Fig. 5, the proposed algorithm is robust to the choice of c_n over a wide range (e.g., 10^{-8} to 10^{-6}), whereas excessively large values were found to degrade solution quality by loosening the upper bound.*

Remark 7. *We use c_n in (10) for any iteration index j, but in principle, it can differ across iterations. This means replacing c_n in (10) with $c_n^{(j)} > 0$, and we can choose the hyperparameters $c_n^{(j)}$ for $j \in \{1, 2, \ldots\}$ and $n \in \mathcal{N}$. However, in that case, the results will not be as neat as what we have now in Result "(ii)" of Theorem 2, since $c_n^{(j)}$ depending on j cannot be simply written in the objective function of an optimization problem (i.e., what we do now in $L_{\boldsymbol{c}}(\boldsymbol{x})$ of (12) cannot work if $c_n^{(j)} > 0$ differ for $j \in \{1, 2, \ldots\}$).*

D. Extending Results of Theorems 1 and 2

In this subsection, we extend the results of Theorems 1 and 2 as follows.

1) From multiplicative programming (MP) to fractional programming (FP): Theorems 1 and 2 are for multiplicative programming (MP). Clearly, multiplicative term(s) can be easily written as fractional terms via a change of variables. Taking Theorem 1 as an example, we can obtain the corresponding result for FP after replacing $A_n(\boldsymbol{x})$ with $\frac{1}{D_n(\boldsymbol{x})}$ for $D_n(\boldsymbol{x}) : \mathbb{R}^M \to \mathbb{R}_{++}$ at necessary places, as elaborated below:

- replacing $K_n(\boldsymbol{x}, y_n)$ of (2) by $K_n^{\mathrm{F}}(\boldsymbol{x}, y_n) \stackrel{\text{def}}{=} \frac{y_n}{[D_n(\boldsymbol{x})]^2} + \frac{[B_n(\boldsymbol{x})]^2}{4y_n}$, where $\boldsymbol{x} \in \mathcal{X}, y_n \in \mathbb{R}_{++}$ and the superscript "F" means "fractional",
- replacing $H(\boldsymbol{x})$ of (4) by $H^{\mathrm{F}}(\boldsymbol{x}) \stackrel{\text{def}}{=} G(\boldsymbol{x}) + \sum_{n=1}^{N} \frac{B_n(\boldsymbol{x})}{D_n(\boldsymbol{x})}$,
- replacing $y_n^{(j)} \stackrel{\text{def}}{=} \frac{B_n(\boldsymbol{x}^{(j-1)})}{2A_n(\boldsymbol{x}^{(j-1)})}$ in (7) by $y_n^{(j)} \stackrel{\text{def}}{=} \frac{D_n(\boldsymbol{x}^{(j-1)})B_n(\boldsymbol{x}^{(j-1)})}{2}$.

For Theorem 2, we can obtain the corresponding result for FP after replacing $A_n(\boldsymbol{x})$ with $\frac{1}{D_n(\boldsymbol{x})}$.

2) Applicability to general optimization problems involving multiplicative or fractional terms: Although Theorems 1 and 2 are about the *sum* of multiplicative term(s) $A_n(\boldsymbol{x})B_n(\boldsymbol{x})$ (after ignoring the "$G(\boldsymbol{x})$" part of (5)), the approach presented in Theorems 1 and 2 can be extended to beyond optimizing the sum. Below, we take Theorem 1 as an example for elaboration and omit the corresponding discussion for Theorem 2. Actually, for any optimization problem $\mathbb{P}$ (with the variable being a vector $\boldsymbol{x}$) involving multiplicative term(s), we can replace any multiplicative term $A_n(\boldsymbol{x})B_n(\boldsymbol{x})$ by $K_n(\boldsymbol{x}, y_n)$ defined in (2), as long as such substitution results in an upper bound of the objective function in minimization problems or a lower bound of the objective function in maximization problems. Then we can solve the resulting optimization problem $\mathbb{Q}$ with the variables being $\boldsymbol{x}$ and $\boldsymbol{y}$. Similarly, for any optimization problem $\mathbb{P}$ involving fractional term(s), we can replace any fractional term $\frac{B_n(\boldsymbol{x})}{D_n(\boldsymbol{x})}$ by $K_n^{\mathrm{F}}(\boldsymbol{x}, y_n)$ defined in Section II-D1. To see the above concretely, a specific example is as follows. For the problem $\mathbb{P}$ of minimizing $G(\boldsymbol{x}) + f_n\big(\sum_{n=1}^{N}[A_n(\boldsymbol{x})B_n(\boldsymbol{x})]\big)$ subject to $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^M$ for non-decreasing functions f_n, we can consider the problem $\mathbb{Q}$ of minimizing $Z(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} G(\boldsymbol{x}) + f_n\big(\sum_{n=1}^{N} K_n(\boldsymbol{x}, y_n)\big)$, where $K_n(\boldsymbol{x}, y_n)$ is defined in (2). In view of the alternating optimization (AO) method in Theorem 1, we also use AO to solve problem $\mathbb{Q}$. Specifically, with an initial $\boldsymbol{x}^{(0)} \in \mathcal{X}$, we perform the following two operations iteratively to get $\boldsymbol{y}^{(j)}$ and $\boldsymbol{x}^{(j)}$ for $j = 1, 2, \ldots$:

- use (7) to set $\boldsymbol{y}^{(j)}$, which optimally solves $\mathbb{Q}(\boldsymbol{x}^{(j-1)})$), with $\mathbb{Q}(\boldsymbol{x})$ denoting optimizing $\boldsymbol{y}$ for $\mathbb{Q}$ given $\boldsymbol{x}$;
- solve Problem $\mathbb{Q}(\boldsymbol{y}^{(j)})$ to obtain $\boldsymbol{x}^{(j)}$ as a KKT point of $\mathbb{Q}(\boldsymbol{y}^{(j)})$, where $\mathbb{Q}(\boldsymbol{y})$ means optimizing $\boldsymbol{x}$ for $\mathbb{Q}$ given $\boldsymbol{y}$.

Then similar to the proof of Theorem 1, $(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})$ converges as $j \to \infty$, and with $(\boldsymbol{x}^{\diamond}, \boldsymbol{y}^{\diamond})$ denoting its convergence, $\boldsymbol{x}^{\diamond}$ is a KKT point for Problem $\mathbb{P}$.

III. Application Scenarios

In this section, we delve into two significant practical applications of the UP transform in wireless communication networks. In subsection III-A, we consider the partial offloading scenario in wireless networking and discuss the problems with continuous optimization variables. In subsection III-B, we investigate the strategies for efficient resource and service allocation across the hierarchical network [15], and focus on the scenario concerning mixed-integer optimization (MIO) mentioned in **Challenge 2**. A pivotal contribution of this application scenario in mixed discrete-continuous cases is the development of an innovative algorithm designed to tackle binary variables (mentioned in *Remark* 4) by adopting our proposed transform. The specific parameter settings and experimental evaluations of these application scenarios are shown in Section III-C.

A. Scenario 1: Partial Offloading with Resource Allocation

Task offloading in mobile edge computing (MEC) has been extensively studied as an effective means in the research of federated learning [16], mobile computing [17], and the Internet of Things (IoT) [18] to jointly reduce computation latency and energy consumption of mobile devices [19]. Existing works typically consider either binary [20] or partial task offloading [21] schemes and formulate joint optimization problems involving offloading decisions, communication resources, and computation resources at both mobile devices and edge servers. Due to the inherent coupling among these variables, most prior studies resort to block coordinate descent (BCD) [22], [23], successive convex approximation (SCA) techniques [22], [24], or deep reinforcement learning (DRL) [25]–[27] to obtain optimal solutions. However, such approaches often suffer from slow convergence or strong sensitivity to variable updating order, especially when the objective function contains multiplicative terms arising from the interaction between offloading ratios and resource allocation variables.

In contrast to these existing studies, our work in this scenario introduces a UP transform-based framework that systematically handles the multiplicative structure in MEC task offloading problems. By reformulating the original non-convex objective into an equivalent surrogate with introduced auxiliary variables, the proposed approach enables a principled two-block optimization structure and admits closed-form updates via KKT analysis.

System Model: To formulate the optimization problem from practical application scenarios, we consider a general mobile edge computing system with N user equipment (UEs) and one edge server. Assume that each UE n processes a computation-intensive task that commonly arises in the vehicular edge computing (VEC), Metaverse [28], or encryption computation, and the size of the task is denoted as C_n. In order to enhance spectral efficiency and minimize cross-channel interference, OFDMA (Orthogonal Frequency-Division Multiple Access) is adopted in this proposed system. Due to computational limitations, each UE offloads a fraction $x_n \in [0, 1]$ of its tasks. $x_n = 0$ indicates the UE n's task will be processed locally on the mobile device, while $x_n = 1$ indicates that the computation task is fully offloaded to the edge server and processed. When $0 < x_n < 1$, it indicates the proportion of tasks that are offloaded to the edge server. By denoting the number of CPU cycles required to process task one bit in the local device and edge server as q_n^{local} and q_n^{edge}, we can then obtain the delay for UE n's computation task at different points as

$$T_n(x_n, f_n^l, f_n^e) = T_n^{\text{local}}(x_n, f_n^l) + T_n^{\text{edge}}(x_n, f_n^e). \tag{13}$$

Specifically, $T_n^{\text{local}}(x_n, f_n^l) = \frac{C_n^{\text{local}} q_n^{\text{local}}}{f_n^l} = \frac{(1-x_n)C_n q_n^{\text{local}}}{f_n^l}$, and $T_n^{\text{edge}}(x_n, f_n^e) = \frac{C_n^{\text{edge}} q_n^{\text{edge}}}{f_n^e} = \frac{x_n C_n q_n^{\text{edge}}}{f_n^e}$, where f_n^l and f_n^e represent the computation frequency of UE n and edge server respectively. Compared with the latency of processing, we omit transmission latency due to the proximity of the edge server to mobile devices.

Therefore, the total energy consumption is derived as $E_n(x_n, f_n^l, f_n^e) = k_n T_n^{\text{local}}(x_n, f_n^l) {f_n^l}^3 + k_e T_n^{\text{edge}}(x_n, f_n^e) {f_n^e}^3$, where k_n is the coefficient reflecting the power efficiency of UE n and k_e represents the analogous coefficient related to the power efficiency of the edge server.

We next infer the cost function which jointly considers the offloading ratio and power limitation of each UE as $O_n(x_n, f_n^l, f_n^e) = w_1 T_n(x_n, f_n^l, f_n^e) + w_2 E_n(x_n, f_n^l, f_n^e)$ and after mathematical transformation, it can be expressed as:

$$O_n(x_n, f_n^l, f_n^e) = (1 - x_n) H_{n,1}(f_n^l) + x_n H_{n,2}(f_n^e),$$

where $H_{n,1}(f_n^l) = C_n q_n^{\text{local}} (\frac{w_1}{f_n^l} + w_2 k_n {f_n^l}^2)$ and $H_{n,2}(f_n^e) = C_n q_n^{\text{edge}} (\frac{w_1}{f_n^e} + w_2 k_e {f_n^e}^2)$. w_1 and w_2 serve as weight parameters modulating the magnitudes of the cost components.

Problem Formulation: UE n can choose the offloading decision $\boldsymbol{x} \coloneqq (x_1, x_2, ..., x_N)$ based on its own and the edge server's computational ability. Meanwhile, the task computation frequency locally $\boldsymbol{f^l} \coloneqq (f_1^l, f_2^l, ..., f_N^l)$, the edge computation frequency $\boldsymbol{f^e} \coloneqq (f_1^e, f_2^e, ..., f_N^e)$ allocated to each mobile equipment can be jointly optimized to achieve optimum. Then, the joint optimization problem incorporating energy cost expenditure, offloading determinations, and computation resources is formulated as follows:

$$\underset{\boldsymbol{x},\ \boldsymbol{f^l},\ \boldsymbol{f^e}}{\text{minimize}} \quad \sum_{n \in \mathcal{N}} (1 - x_n) H_{n,1}(f_n^l) + x_n H_{n,2}(f_n^e) \tag{14}$$

$$\text{subject to} \quad (\mathbf{C}_1)\text{: } 0 \le x_n \le 1, \forall n \in \mathcal{N}, \tag{14a}$$

$$(\mathbf{C}_2)\text{: } \sum\nolimits_{n=1}^{N} f_n^e \le F^e, \tag{14b}$$

$$(\mathbf{C}_3)\text{: } 0 \le f_n^l \le F_n^l, \forall n \in \mathcal{N}, \tag{14c}$$

$$(\mathbf{C}_4)\text{: } 0 \le f_n^e \le F_n^e, \forall n \in \mathcal{N}, \tag{14d}$$

where $\mathcal{N} = \{1, 2, ..., N\}$ is the set of UEs. F^e in $(\mathbf{C}_2)$ and F_n^l in $(\mathbf{C}_3)$ indicate the maximum computation frequency of the edge server and UE n. F_n^e in $(\mathbf{C}_4)$ represented the maximum allocated computation frequency by the edge server to UE n. For each constraint, $(\mathbf{C}_1)$ indicates the offloading decision and the ratio of the offloaded computation task of each UE. $(\mathbf{C}_2)$ restricts the total computation frequency of the edge server. $(\mathbf{C}_3)$ represents that the local processing frequency cannot exceed the device limit. For constraint $(\mathbf{C}_4)$, it mitigates the potential for specific computation tasks to greedily consume computational resources on the edge server.

Algorithm 1: Partial Offloading Programming

Initialize the index of iteration: $k=0$; optimization variable $\boldsymbol{\mathcal{X}}^{(0)}=[\boldsymbol{x}^{(0)}, \boldsymbol{f^{l}}^{(0)}, \boldsymbol{f^{e}}^{(0)}]$;
Calculate and derive auxiliary variable space: $\boldsymbol{\mathcal{A}}^{(0)}=[\boldsymbol{u}^{(0)}, \boldsymbol{v}^{(0)}]$, where $u_n^{(0)}=\frac{(1-x_n^{(0)})}{2H_{n,1}(f_n^{l\,(0)})}$, $v_n^{(k)}=\frac{x_n^{(0)}}{2H_{n,2}(f_n^{e\,(0)})}$, $\forall n\in\mathcal{N}$; Derive the constant vector: $\boldsymbol{c}^{(0)}$ based on (10).
repeat
  Obtain the optimal variable $\boldsymbol{\mathcal{X}}^{(k+1)}$ of the $(k+1)$-th iteration by adopting the Algorithm 2 when given auxiliary variable space $\boldsymbol{\mathcal{A}}^{(k)}$;
  Update $\boldsymbol{\mathcal{A}}^{(k+1)}=[\boldsymbol{u}^{(k+1)}, \boldsymbol{v}^{(k+1)}]$ and $\boldsymbol{c}^{(k+1)}$ with given $\boldsymbol{\mathcal{X}}^{(k+1)}$;
  $k\leftarrow k+1$;
until *Convergence or Maximum iteration number* K;

Solution with UP Transform: Based on Theorem 2, the original optimization is divided into the sub-problem and the coupling of the optimization variable $\boldsymbol{x}$, $\boldsymbol{f^l}$, and $\boldsymbol{f^e}$ can be resolved by introducing auxiliary variables in (7). In the $(k+1)$-th iteration, the objective function is approximated as:

$$G(\boldsymbol{x}, \boldsymbol{f^l}, \boldsymbol{f^e}|\boldsymbol{u}^{(k)}, \boldsymbol{v}^{(k)}, \boldsymbol{c}^{(k)})=\sum_{n\in\mathcal{N}} g_n(x_n, f_n^l, f_n^e|u_n^{(k)}, v_n^{(k)}, c_n^{(k)}),$$

where $u_n^{(k)}=\frac{(1-x_n^{(k)})}{2H_{n,1}(f_n^{l\,(k)})}$ and $v_n^{(k)}=\frac{x_n^{(k)}}{2H_{n,2}(f_n^{e})^{(k)}}\in\mathbb{R}_+$ are the introduced auxiliary variables related to the convergence of the proposed algorithm when $0<x_n<1$ according to (7). Function $g_n(x_n, f_n^l, f_n^e \mid u_n^{(k)}, v_n^{(k)})$ is defined as shown below, where $c_n^{(k)}$ is the introduced constant according to (10):

$$g_n(x_n, f_n^l, f_n^e|u_n^{(k)}, v_n^{(k)})=[H_{n,1}(f_n^l)]^2(u_n^{(k)}+c_n^{(k)})+\frac{(1-x_n)^2}{4\left(u_n^{(k)}+c_n^{(k)}\right)}$$
$$+[H_{n,2}(f_n^e)]^2(v_n^{(k)}+c_n^{(k)})+\frac{x_n^2}{4(v_n^{(k)}+c_n^{(k)})} \quad (15)$$

Then sub-problem of the original optimization problem in the $(k+1)$-th iteration is reformulated as:

$$\underset{\boldsymbol{x},\ \boldsymbol{f^l},\ \boldsymbol{f^e}}{\text{minimize}}\ G(\boldsymbol{x}, \boldsymbol{f^l}, \boldsymbol{f^e}\mid \boldsymbol{u}^{(k)}, \boldsymbol{v}^{(k)}, \boldsymbol{c}^{(k)}) \quad (16)$$
$$\text{subject to (14a), (14b), (14c), (14d).}$$

Until now, the original optimization problem can be solved iteratively, and the process of solution is listed in Algorithm 1. Instead of using the general convex toolboxes, we adopt Karush–Kuhn–Tucker (KKT) conditions analysis to obtain the solution to the problem in a faster way.

KKT Analysis: Before proceeding with the analysis of each KKT condition, we first define the Lagrange function by introducing multipliers for constraints as:

$$L=G(\boldsymbol{x}, \boldsymbol{f^l}, \boldsymbol{f^e}|\boldsymbol{u}^{(k)}, \boldsymbol{v}^{(k)}, \boldsymbol{c}^{(k)})+\sum_{n\in\mathcal{N}}[\gamma_n\cdot(x_n-1)-\beta_n x_n]+\delta\cdot[(\sum_{n\in\mathcal{N}} f_n^e)$$
$$-F^e]+\sum_{n\in\mathcal{N}}[\zeta_n\cdot(f_n^l-F_n^l)-\epsilon_n f_n^l]+\sum_{n\in\mathcal{N}}[\theta_n\cdot(f_n^e-F_n^e)-\eta_n f_n^e] \quad (17)$$

Then, based on the multipliers and Lagrange function, we get KKT conditions as shown below:

Stationary:

$$\frac{\partial L}{\partial x_n}=D_n(x_n)-\beta_n+\gamma_n=0, \quad (18a)$$
$$\frac{\partial L}{\partial f_n^l}=Q_n(f_n^l)-\epsilon_n+\zeta_n=0, \quad (18b)$$

Algorithm 2: Analyze Corresponding KKT Conditions

Given the auxiliary variable space $\boldsymbol{\mathcal{A}}=[\boldsymbol{u}, \boldsymbol{v}]$;
for $n\leftarrow 1$ to N **do**
  Obtain $\widetilde{x}_n$ by assuming $D_n(x_n)=0$ on (18a);
  Obtain $\widetilde{f}_n^l$ by assuming $Q_n(f_n^l)=0$ on (18b);
  Obtain $\widetilde{f}_n^e(\delta)$ assuming $R_n(f_n^l, \delta)=0$ on (19c);
end for
$\delta^*\leftarrow\begin{cases}0, \text{if}\sum_{n\in\mathcal{N}}\widetilde{f}_n^e(0)\le F^e;\\ \text{Solution to } \sum_{n\in\mathcal{N}}\widetilde{f}_n^e(0)>F^e;\end{cases}$
update x_n^*, ${f_n^l}^*$, and f_n^{e*} based on Theorem 3;
return The optimal variable value $\mathcal{X}^*=[\boldsymbol{x}^*, \boldsymbol{f^l}^*, \boldsymbol{f^e}^*]$.

$$\frac{\partial L}{\partial f_n^e}=R_n(f_n^e, \delta)-\eta_n+\theta_n=0. \quad (18c)$$

where $D_n(x_n)=\frac{x_n-1}{2(u_n^{(k)}+c_n^{(k)})}+\frac{x_n}{2(v_n^{(k)}+c_n^{(k)})}$ relating to the variable x_n, $Q_n(f_n^l)=2H_{n,1}(f_n^l)H_{n,1}^{'}(f_n^l)(u_n^{(k)}+c_n^{(k)})$ relating to the variable f_n^l, and $R_n(f_n^e, \delta)=2H_{n,2}(f_n^e)H_{n,2}^{'}(f_n^e)(v_n^{(k)}+c_n^{(k)})+\delta$ relating to the variable f_n^e and the multiplier δ.

Complementary Slackness:

$$\begin{aligned}&\text{(19a): } \beta_n\cdot(-x_n)=0; \quad \text{(19b): } \gamma_n\cdot(x_n-1)=0;\\ &\text{(19c): } \delta\cdot[(\sum\nolimits_{n\in\mathcal{N}} f_n^e)-F^e]=0;\\ &\text{(19d): } \epsilon_n\cdot(-f_n^l)=0; \quad \text{(19e): } \zeta_n\cdot(f_n^l-F_n^l)=0;\\ &\text{(19f): } \eta_n\cdot(-f_n^e)=0; \quad \text{(19g): } \theta_n\cdot(f_n^e-F_n^e)=0;\end{aligned} \quad (19)$$

Primal Feasibility: (14a), (14b), (14c), (14d);

Dual Feasibility:

$$\text{(20a)-(20e)}: \beta_n, \gamma_n, \delta, \epsilon_n, \zeta_n, \eta_n, \theta_n\ge 0, \forall n\in\mathcal{N}. \quad (20)$$

Under the KKT conditions, we seek the optimal solutions by employing the proposed algorithm listed in Algorithm 2.

Theorem 3. *The optimal solution of the proposed objective function is obtained by* ***Algorithm*** *2 and is expressed as:*

$$\begin{cases} x_n^*=\max\{\min\{\widetilde{x}_n, 1\}, 0\};\\ {f_n^l}^*=\max\{\min\{\widetilde{f}_n^l, F_n^l\}, 0\};\\ f_n^{e*}=\max\{\min\{\widehat{f}_n^e(\delta^*), F_n^e\}, 0\};\end{cases} \quad (21)$$

where $\widetilde{x}_n$ meets the condition (18a) with $D_n(x_n)|_{x_n=\widetilde{x}_n}=0$, $\widetilde{f}_n^l$ and $\widehat{f}_n^e(\delta)$ meets the condition (18b) and (18c) with $Q_n(f_n^l)|_{f_n^l=\widetilde{f}_n^l}=0$ and $R_n(f_n^e, \delta)|_{f_n^e=\widehat{f}_n^e(\delta)}=0$, respectively.

Proof: Please refer to Appendix C. □

B. Scenario 2: User Association with Resource Allocation

In this subsection, we consider a heterogeneous mobile edge computing system consisting of one Macro Base Station (MBS) and one Small Base Station (SBS) [29], [30] as an example and propose a novel two-tier computation offloading algorithm with the convergence of the UP transform and successive convex approximation (SCA) approach [31].

System Model: In the proposed system, we assume that each base station also has one MEC server to execute the offloaded computation task. The network comprises N mobile users $\mathcal{N}=\{1,\ldots,N\}$ and base stations $\mathcal{M}=\{1,2\}$, where $m=1$ denotes the SBS and $m=2$ the MBS. The SBS connects to the MBS via wired links, and users are distributed within the MBS coverage. Each user n is connected to the SBS via wireless links and has a task $D_n=(d_n, c_n)$, defined by data size d_n and CPU cycles per bit c_n.

For this two-tier computation offloading framework, a mobile user can partition the task into two parts. One part is offloaded to the MEC server which is associated with the base stations, and the other part is executed locally. As the computation resource of the MEC server in SBS is limited, the part offloaded will further be partitioned into two parts when the computation resource of the MEC server on SBS is exhausted. The SBS will offload the remaining part of the task to the MEC server on the MBS. Furthermore, each mobile user can also offload the task directly to the MBS. We denote the part offloaded to the base stations as $d_{n,m}(0 \leq d_{n,m} \leq d_n)$ and the part processed locally as $d_n - d_{n,m}$. In the second tier, SBS further divides the fragment $d_{n,1}$ into $d_{n,1} - d_{n,1}^{'}$ and $d_{n,1}^{'}(0 \leq d_{n,1}^{'} \leq d_{n,1} \leq d_n)$, and the latter part is offloaded to the MBS. For the offloading decision of each user, we denote $x_{n,m} = \{0, 1\}$ as the association variable, where $x_{n,m} = 0$ if user $n \in \mathcal{N}$ is associated with base station $m \in \mathcal{M}$ and $x_{n,m} = 1$ otherwise.

For the wireless communication model, we also adopt the OFDMA technique between mobile users and base stations, and the channel power gain is denoted as $H_{n,m}$. Then, the uplink SINR is $\gamma_{n,m} = \frac{P_n H_{n,m}}{\sigma^2}$ $(m \in \{1, 2\})$, where P_n is transmit power and σ^2 is noise power. The inter-user interference is not taken into consideration with the adoption of OFDMA. Assuming equal allocation of bandwidth B_m among associated users, the data rates for SBS and MBS are $R_{n,1} = \frac{B_1}{\sum_{n=1}^{N} x_{n,1}} \ln(1 + \gamma_{n,1})$ and $R_{n,2} = \frac{B_2}{\sum_{n=1}^{N} x_{n,2}} \ln(1 + \gamma_{n,2})$, respectively, where $\sum_{n=1}^{N} x_{n,m}$ represents the user load.

Then, we further infer the cost function of the local computation in terms of latency and energy consumption. We denote f_n^l as the computation resource when processing the task locally. Then we obtain the computation time as $T_{n,m}^{\text{local}} = \frac{(d_n - d_{n,m})c_n}{f_n^l}$, and the corresponding energy consumption is $E_{n,m}^{\text{local}} = k^l (d_n - d_{n,m}) c_n {f_n^l}^2$ where k^l is the effective switched capacitance depending on the chip architecture.

For the task offloaded to the SBS, we denote the allocated computation frequency by the SBS to the task as f_n^s and assume that MBS allocates fixed computation frequency f_0 to each offloaded task. Then the processing time in SBS is:

$$T_{n,1}^{\text{SBS}} = \frac{d_{n,1}}{R_{n,1}} + \frac{(d_{n,1} - d_{n,1}^{'})c_n}{f_n^s} + \frac{d_{n,1}^{'}}{r_0} + \frac{d_{n,1}^{'} c_n}{f_0},$$

where r_0 represents the data rate of the wired link. Then, the corresponding energy consumption is derived below:

$$E_{n,1}^{\text{SBS}} = \frac{P_n d_{n,1}}{R_{n,1}} + k^S (d_{n,1} - d_{n,1}^{'}) c_n {f_n^s}^2 + \frac{\bar{P} d_{n,1}^{'}}{r_0} + k^M d_{n,1}^{'} c_n f_0^2,$$

where $\bar{P}$ is the offloading power via wired line, k^S and k^M are the coefficients reflecting the power efficiency of the MEC server in SBS and MBS, respectively. When the mobile user directly offloads the task to the MEC server in MBS, the execution time is $T_{n,2}^{\text{MBS}} = \frac{d_{n,2}}{R_{n,2}} + \frac{d_{n,2} c_n}{f_0}$, and the energy consumption is $E_{n,2}^{\text{MBS}} = \frac{P_n d_{n,2}}{R_{n,2}} + k^M d_{n,2} c_n f_0^2$.

Then, the total latency and energy consumption of the task for each user can be concluded as follows:

$$T_n = \left(\sum\nolimits_{m \in \mathcal{M}} x_{n,m} T_{n,m}^{\text{local}}\right) + x_{n,1} T_{n,1}^{\text{SBS}} + x_{n,2} T_{n,2}^{\text{MBS}},$$

$$E_n = \left(\sum\nolimits_{m \in \mathcal{M}} x_{n,m} E_{n,m}^{\text{local}}\right) + x_{n,1} E_{n,1}^{\text{SBS}} + x_{n,2} E_{n,2}^{\text{MBS}}.$$

Problem Formulation: We define $\boldsymbol{x} \coloneqq \{x_{n,m}\}$ as the user association decision vector, $\boldsymbol{f^l} \coloneqq \{f_n^l\}$ and $\boldsymbol{f_n^s} \coloneqq \{f_n^s\}$ as the vector of local and SBS computation frequency respectively, $\boldsymbol{P} \coloneqq [P_n|_{n \in \mathcal{N}}]$ as the transmission power vector, and $\boldsymbol{d} \coloneqq \{d_{n,m}, d_{n,1}^{'}\}$ as the computation offloading vector. The optimization problem is formulated as:

$$\underset{\boldsymbol{x}, \boldsymbol{f^l}, \boldsymbol{f^s}, \boldsymbol{P}, \boldsymbol{d}}{\text{minimize}} \quad \sum_{n=1}^{N} (w_1 T_n + w_2 E_n) \tag{22}$$

$$\text{subject to} \quad (\mathbf{C}_1) \sum_{m \in \mathcal{M}} x_{n,m} = 1, \forall n \in \mathcal{N}, \tag{22a}$$

$$(\mathbf{C}_2) \quad 0 \leq f_n^l \leq F_n, \forall n \in \mathcal{N}, \tag{22b}$$

$$(\mathbf{C}_3) \quad \sum_{n=1}^{N} x_{n,1} f_n^s \leq F^s, \tag{22c}$$

$$(\mathbf{C}_4) \quad 0 \leq P_n \leq P_n^{\max}, \forall n \in \mathcal{N}, \tag{22d}$$

$$(\mathbf{C}_5) \quad 0 \leq d_{n,1}^{'} \leq d_{n,1}, \forall n \in \mathcal{N}, \tag{22e}$$

$$(\mathbf{C}_6) \quad 0 \leq d_{n,m} \leq d_n, \forall n \in \mathcal{N}, m \in \mathcal{M}, \tag{22f}$$

$$(\mathbf{C}_7) \quad f_n^s \geq 0, \forall n \in \mathcal{N}, \tag{22g}$$

$$(\mathbf{C}_8) \quad x_{n,m} \in \{0, 1\}, \forall n \in \mathcal{N}, m \in \mathcal{M}. \tag{22h}$$

where w_1 and w_2 represent the weight parameters designed to modulate the magnitudes of the cost components. The offloading decision constraints relate to constrains $(\mathbf{C}_1)$, $(\mathbf{C}_3)$ and $(\mathbf{C}_8)$. Constraints $(\mathbf{C}_2)$, $(\mathbf{C}_3)$, and $(\mathbf{C}_7)$ limit the frequency of computation locally and in SBS, respectively. Also, constraints $(\mathbf{C}_5)$ and $(\mathbf{C}_6)$ also constructed as the limitation of computation resource. $(\mathbf{C}_4)$ limits the power resource of mobile user n.

Solution with Updated Transform and SCA: Based on the constraints of the user association vector, we first adopt SCA technology to solve the discrete optimization variable $\boldsymbol{x}$ resulting in the problem as NP-hard. Without loss of equivalence, $(\mathbf{C}_8)$ can be rewritten as:

$$x_{n,m} \in [0, 1], \ \forall n \in \mathcal{N}, m \in \mathcal{M}, \tag{23}$$

$$\textbf{and} \ \sum\nolimits_{n \in \mathcal{N}} \sum\nolimits_{m \in \mathcal{M}} x_{n,m}(1 - x_{n,m}) \leq 0. \tag{24}$$

Note that the optimization problem has been transitioned into a continuous optimization problem, resulting in a notable reduction in computational complexity when contrasted with the direct resolution of the original discrete variable $x_{n,m}$. However, the function $\sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}} x_{n,m}(1 - x_{n,m})$ in constraint (24) is a concave function. To facilitate the solution, we adopt a method that introduces a penalty term for this concave constraint into the objective function, which is expressed as:

$$\sum_{n=1}^{N} (w_1 T_n + w_2 E_n) - \tau \cdot \sum\nolimits_{n \in \mathcal{N}} \sum\nolimits_{m \in \mathcal{M}} x_{n,m}(x_{n,m} - 1),$$

where τ is the penalty parameter with $\tau > 0$. Then, the objective function becomes concave due to the concavity of the second term. Simultaneously, given the second term is differentiable, we utilize the first-order Taylor series to linearize it at each iteration. Specifically, in the $(i+1)$-th iteration,we approximate each $\sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}} x_{n,m}(x_{n,m} - 1)$ with $\sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}} x_{n,m}^{(i)}(x_{n,m}^{(i)} - 1) + (2x_{n,m}^{(i)} - 1)(x_{n,m} - x_{n,m}^{(i)})$ denoted as $H(\boldsymbol{x} \mid \boldsymbol{x}^{(i)})$, where $\boldsymbol{x}^{(i)}$ is the optimal solution of the i-th sub-problem. Therefore, the objective

function is reformulated as:

$$\sum_{n=1}^{N}(w_1 T_n + w_2 E_n) - \tau \cdot H(\boldsymbol{x} \mid \boldsymbol{x}^{(i)}). \tag{25}$$

After transformation, we obtain the objective function as:

$$\sum_{n=1}^{N} Q_n(\boldsymbol{x}, \boldsymbol{f^l}, \boldsymbol{f^s}, \boldsymbol{P}, \boldsymbol{d}) - \tau \cdot H(\boldsymbol{x} \mid \boldsymbol{x}^{(i)}),$$

where $Q_n(\boldsymbol{x}, \boldsymbol{f^l}, \boldsymbol{f^s}, \boldsymbol{P}, \boldsymbol{d}) = O_n(\boldsymbol{x}, \boldsymbol{f^l}, \boldsymbol{d}) + S_n(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{d}) + U_n(\boldsymbol{x}, \boldsymbol{f^s}, \boldsymbol{d}) + V_n(\boldsymbol{x}, \boldsymbol{d})$ and specifically,

$$V_n(\boldsymbol{x}, \boldsymbol{d}) = x_{n,1} d_{n,1}^{'} C + x_{n,2} d_{n,2} (C - \frac{w_1 + w_2 \bar{P}}{r_0}),$$

$$U_n(\boldsymbol{x}, \boldsymbol{f^s}, \boldsymbol{d}) = x_{n,1}(d_{n,1} - d_{n,1}^{'})(\frac{w_1}{f_n^s} + w_2 k^S {f_n^s}^2) c_n,$$

$$O_n(\boldsymbol{x}, \boldsymbol{f^l}, \boldsymbol{d}) = \sum_{m \in \mathcal{M}} x_{n,m}(d_n - d_{n,m})(\frac{w_1}{f_n^l} + w_2 k^l {f_n^l}^2) c_n,$$

$$S_n(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{d}) = \sum_{m \in \mathcal{M}} x_{n,m} \frac{d_{n,m}(\sum_{n=1}^{N} x_{n,m})}{B_m \ln(1 + \frac{P_n H_{n,m}}{\sigma^2})}(w_1 + w_2 P_n),$$

and the constant $C = \frac{w_1 + w_2 \bar{P}}{r_0} + (\frac{w_1}{f_0} + w_2 k^M f_0^2) c_n$.

Theorem 4. *Proposed objective function $Q_n(\boldsymbol{x}, \boldsymbol{f^l}, \boldsymbol{f^s}, \boldsymbol{P}, \boldsymbol{d})$ with coupled optimization variable can solved by optimizing convex function $\widetilde{Q}_n(\boldsymbol{\mathcal{X}} \mid \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}})$ derived from the UP transform, where $\mathcal{X} \coloneqq [\boldsymbol{x}, \boldsymbol{f^l}, \boldsymbol{f^s}, \boldsymbol{P}, \boldsymbol{d}]$ is the space of optimization variable, $\boldsymbol{\mathcal{A}} \coloneqq [\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\epsilon}, \boldsymbol{\zeta}, \boldsymbol{\eta}, \boldsymbol{\theta}]$ is the auxiliary variables space, and $\boldsymbol{\mathcal{B}} \coloneqq [\boldsymbol{b}, \boldsymbol{e}, \boldsymbol{g}, \boldsymbol{h}, \boldsymbol{j}, \boldsymbol{l}, \boldsymbol{q}, \boldsymbol{z}]$ is the constant space.*

Proof: Please refer to Appendix D. □

Then the transformed objective function can be derived as: $\sum_{n=1}^{N} \widetilde{Q}_n(\boldsymbol{\mathcal{X}} \mid \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}) - \tau \cdot H(\boldsymbol{x} \mid \boldsymbol{x}^{(i)})$, where $\widetilde{Q}_n(\boldsymbol{\mathcal{X}} \mid \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}) = \widetilde{O}_n + \widetilde{S}_n + \widetilde{U}_n + \widetilde{V}_n$ is given in Appendix D.

However, this optimization problem is not convex as the optimization variable is still coupled in the constraint (22c). We apply the proposed UP transform to decouple the variables:

$$\sum_{n=1}^{N} \Big[(f_n^s)^2 (\lambda_n^{(k)} + u_n^{(k)}) + \frac{x_{n,1}^2}{4(\lambda_n^{(k)} + u_n^{(k)})} \Big] - F^s \leq 0, \tag{26}$$

where auxiliary variable $\lambda_n^{(k)}$ and constant $u_n^{(k)}$ obey the rules in Theorem 2. And $\lambda_n^{(k)} = \frac{x_{n,1}^{(k)}}{2 f_n^{s(k)}}$ in the $(k+1)$-th iteration.

The the sub-problem of the original problem in the $(k+1)$-th iteration of UP transform under the $(i+1)$-th iteration of SCA approach is reformulated as:

$$\underset{\boldsymbol{x}, \boldsymbol{f^l}, \boldsymbol{f^s}, \boldsymbol{P}, \boldsymbol{d}}{\text{minimize}} \quad \sum_{n=1}^{N} \widetilde{Q}_n(\boldsymbol{\mathcal{X}} | \boldsymbol{\mathcal{A}}^{(k)}, \boldsymbol{\mathcal{B}}^{(k)}) - \tau \cdot H(\boldsymbol{x} | \boldsymbol{x}^{(i)}) \tag{27}$$

$$\text{subject to} \quad (22a), (22b), (22d), (22e), (22f), (22g), (23), (26)$$

Then, the original optimization problem can be addressed through an iterative process. For this user association practical application, various analysis methods can be considered. To prevent unnecessary repetition in analyses, the CVX toolboxes are utilized for solving each convex sub-problem efficiently.

Proposed Algorithm: In the proposed method with the convergence of SCA technique, a penalty function is employed to further facilitate the transformation of various conditions, effectively segmenting the algorithm into two complementary components: *Inter-Sub-Problem Programming* and *Intra-Sub-Problem Programming*. For the inter-sub-problem programming as illustrated in Algorithm 3, it focused on deriving the optimal solution $\boldsymbol{x}^{(i)}$ in the i-th sub-problem. This solution is subsequently utilized in the succeeding iteration of the SCA method. Then the proposed intra-sub-problem programming, as illustrated in Algorithm 4, which is obtained after adopting the UP transform, will be utilized to derive the optimal solution in the succeeding k-th iteration. SCA progressively approximates feasible convex subsets within the non-convex space, while the UP transform addresses instability issues caused by multiplicative terms. Their integration ensures feasibility and convergence under non-convex scenarios. Therefore, the proposed algorithm will converge and get the optimal solutions.

Algorithm 3: Inter-Sub-Problem Programming

```
Initialization of the iteration index: i = 1;
Initialization of the optimal solution space:
  S^(0) = [x^(0), f^l(0), f^s(0), P^(0), d^(0)];
Adopt the SCA method to obtain the problem (27);
repeat
    Obtain the i-th sub-problem by using x^(i-1);
    Solve the i-th sub-problem by using Algorithm 4
      to get the optimal optimal variable value
      X* = [x*, f^l*, f^s*, P*, d*];
    Let S^i ← X* of i-th sub-problem;
    Set i ← i + 1;
until |S^(i) − S^(i−1)| ≤ ε̄_0 or Maximum iteration
  number I;
return S^(i) = [x^(i), f^l(i), f^s(i), P^(i), d^(i)] as the optimal
  solution of the original optimization problem.
```

Algorithm 4: Intra-Sub-Problem Programming

```
Initialization of the iteration index: k = 0;
Initialization of the optimization variable space:
  X^(0) = [x^(0), f^l(0), f^s(0), P^(0), d^(0)];
Calculate and derive auxiliary variable space: A^(0) and
  constant space: B^(0) based on the analysis of
  Theorem 4;
repeat
    Obtain the optimal variable X^(k+1) of the
      (k+1)-th iteration by adopting CVX toolboxes
      when given auxiliary variable space A^(k) and
      constant B^(k);
    Update A^(k+1) and B^(k+1) with given X^(k+1);
    k ← k + 1;
until Convergence or Maximum iteration number K;
return X* ← X^(k) as the optimal solution of the i-th
  sub-problem.
```

Complexity and Scalability: The proposed algorithm follows a two-layer iterative structure, as summarized in Algorithms 3 and 4. The outer layer involves SCA iterations indexed by i, while the inner layer applies the UP transform indexed by k. In each outer iteration, Algorithm 3 invokes Algorithm 4 to solve a convex sub-problem. The dominant computational cost arises from solving this convex sub-problem using standard convex optimization tools. The number of decision variables scales linearly with the number of user–SBS pairs, i.e., roughly $\mathcal{O}(NM)$, primarily due to the user association variables $x_{n,m}$ and task partitioning variables

$d_{n,m}$. Consequently, the worst-case complexity of solving one convex sub-problem is polynomial, typically $\mathcal{O}((NM)^{3.5})$, whereas the auxiliary variable updates incur only linear complexity $\mathcal{O}(NM)$. Combining both layers, the overall worst-case computational complexity is $\mathcal{O}\big(I\cdot K\cdot(NM)^{3.5}\big)$, where I and K denote the number of SCA and UP-transform iterations, respectively.

Regarding scalability to larger HetNet deployments, the proposed framework demonstrates significant advantages. Although increasing the number of SBSs (M) enlarges the search space, the problem complexity grows polynomially with M. This contrasts significantly with traditional combinatorial approaches for user association, which suffer from exponential complexity scaling (i.e., $\mathcal{O}(2^{NM})$). Furthermore, the proposed framework remains structurally unchanged for larger M, introducing no additional nested iterations or non-convex coupling terms. In practical deployments, scalability is further enhanced by the sparsity of the network topology, as each user typically associates with only a limited subset of candidate SBSs, effectively reducing the problem dimension.

C. Experimental Results

1) Baseline Settings: To evaluate the performance of our proposed UP transform-based algorithm, we compare it with several baselines commonly used in edge computing and task offloading scenarios:

- **Full Offloading**: Each UE offloads its entire computation task to the edge server. The edge server allocates its computation resources to users proportionally.
- **No Offloading**: All tasks are processed locally at the UE side, with computation frequencies constrained by each device's maximum capability.
- **Random Offloading**: For each UE, the offloading ratio is randomly sampled from a uniform distribution in $[0, 1]$, and the computation resources are allocated accordingly.
- **Gradient Descent / Newton's Method / Interior Point Method (IPM) / B&B Method / Heuristic Allocation**: These are classical optimization methods applied to solve the original objective. Computation complexity comparisons are further presented in Section IV-B.

Implementation Details of Comparison Optimization Methods: To ensure reproducibility and a fair comparison, we strictly standardize the implementation procedures and key hyperparameters for all baseline schemes. All methods utilize the same feasible initialization strategy as the proposed algorithm, adopt a maximum iteration limit of $K = 100$, and terminate when the relative improvement of the objective value falls below $\epsilon = 10^{-4}$. To guarantee feasibility, a unified projection and normalization operation is applied to intermediate solutions that violate box or coupled resource constraints. Specifically, for the Gradient Descent method, we employ a Backtracking Line Search based on the Armijo rule to adaptively determine the step size. This adaptive mechanism is crucial for mitigating the numerical instability caused by large gradients from the penalty term. For Newton's Method, to address the potential indefinite Hessian matrix in non-convex regions and ensure numerical stability comparable to our proposed UP transform (where $c_n = 10^{-6}$), we apply a regularization term $\lambda\mathbf{I}$ with $\lambda = 10^{-6}$ to the Hessian matrix during updates. A backtracking line search is also integrated to compute the optimal damping step size. The IPM is implemented within a logarithmic barrier framework. The barrier parameter is initialized at $\mu_0 = 10$ and decays with a factor of 0.1 at each outer iteration. The resulting unconstrained sub-problems are solved via Newton steps until convergence. For the Branch-and-Bound (B&B) method applied to the mixed-integer problems in Scenario 2, we employ a Best-First Search strategy. At each node, the continuous relaxation is solved using an interior-point solver, and the algorithm terminates when the relative optimality gap drops below 10^{-4}. It is worth noting that while B&B is allowed to run to completion to serve as a global performance benchmark, which is illustrated in Fig. 3, its computational runtime is prohibitive. As illustrated in Fig. 8, B&B easily exceeds practical latency constraints even in small-scale networks. Finally, the Heuristic Allocation follows a Max-SINR strategy: each user associates with the base station providing the highest SINR, and computing/communication resources are allocated proportionally based on server capacities.

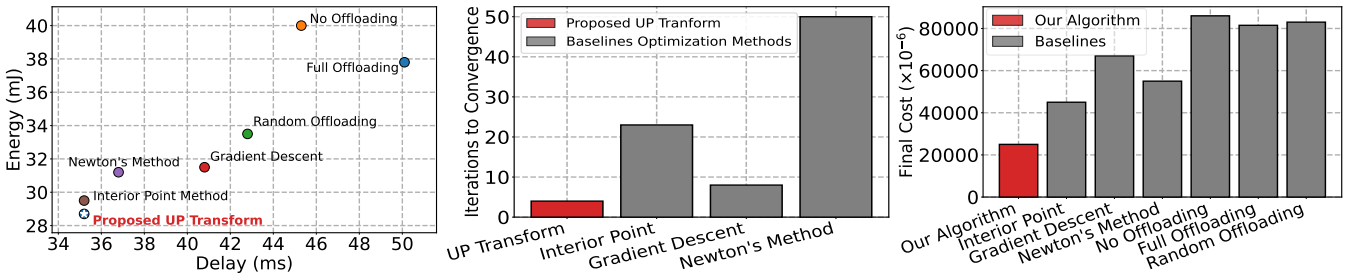
TABLE I: Simulation Parameters for Scenario 1

Parameter	Value
Number of UEs N	30
Edge server computing capacity F^e	10 GHz
Maximum allocable edge CPU per UE F_n^e	10 GHz
Maximum local CPU frequency F_n^l	1.5 GHz
Energy coefficients k_n, k_e	10^{-26}
CPU cycles per bit $q_n^{\text{local}}, q_n^{\text{edge}}$	1000 cycles/bit
Task size C_n	Uniform in $[100, 500]$ MB
Hyperparameter c_n	10^{-6}
Stopping tolerance ε	10^{-4}
Maximum iterations K	100

2) Scenario 1: To evaluate our transform in partial offloading scenarios, overall system parameter settings are summarized in Table I.

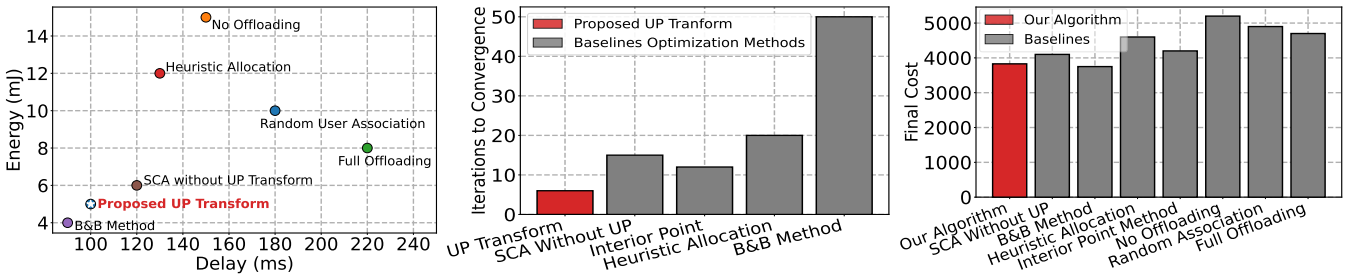
Implementation Details: The optimization variables in this scenario consist of the offloading ratios $\boldsymbol{x}$, local CPU frequencies $\boldsymbol{f}^l$, and edge CPU allocations $\boldsymbol{f}^e$. To facilitate reproducibility, we adopt a feasible initialization strategy: each x_n is randomly initialized in $[0, 1]$; local CPU frequencies f_n^l are randomly sampled from $[0, F_n^l]$; and edge CPU allocations f_n^e are initialized in $[0, F^e]$ and subsequently normalized to satisfy the sum-capacity constraint (14b). Given these initial optimization variables, the auxiliary variables ($\boldsymbol{u}$ and $\boldsymbol{v}$) are computed according to their closed-form expressions derived from the UP transform. regarding the hyperparameters, we set $c_n = 10^{-6}$ for all n. This small positive constant ensures numerical stability when multiplicative terms approach zero, without introducing significant approximation bias. Finally, the algorithm terminates when the relative increase in the objective function falls below a tolerance threshold $\epsilon = 10^{-4}$ or a maximum iteration count $K = 100$ is reached.

Delay-Energy Tradeoff. Fig. 2(a) illustrates the delay-energy tradeoff of baselines. Our algorithm achieves the best balance with minimal energy consumption and short delays, significantly outperforming full and no offloading methods. Random offloading and gradient descent show moderate re-



(a) Delay-Energy Tradeoff (b) Convergence Comparison (c) Final Cost Function Value Comparison

Fig. 2: Evaluation Results of Scenario 1



(a) Delay-Energy Tradeoff (b) Convergence Comparison (c) Final Cost Function Value Comparison

Fig. 3: Evaluation Results of Scenario 2

TABLE II: Simulation Parameters for Scenario 2

Parameter	Value
Number of users N	20
Maximum transmit power $P_n^{\max}$	100 mW
MBS bandwidth	10 MHz
SBS bandwidth	5 MHz
Noise power σ^2	-110 dBm
Path-loss (MBS–UE)	$128.1 + 37.6 \log_{10}(d) + \mu$ dB
Path-loss (SBS–UE)	$140.7 + 37.6 \log_{10}(d) + \mu$ dB
Shadowing μ	$\mathcal{N}(0, 8\,\text{dB})$
SBS computing frequency F^s	20 GHz
User computing frequency F_n	1 GHz
Task size d_n	350 KB
CPU cycles per bit c_n	75 cycles/bit
MBS computing capacity f_0	5 GHz
Wired transmission rate r_0	1 Gbps
Energy coefficients k^l, k^S, k^M	10^{-25}
Penalty parameter τ	10^5
Element of hyperparameter space	10^{-6}
Stopping tolerance ε	10^{-4}
Maximum iterations K	100

sults, while the interior-point and Newton's methods provide competitive delay performance but at higher energy costs.

Convergence Efficiency. Fig. 2(b) compares the convergence speed of traditional methods. Our proposed UP transform-based algorithm requires fewer iterations, significantly outperforming traditional methods (gradient descent, interior-point, and Newton's method) due to its well-designed surrogate function and adaptive auxiliary variables, which accelerate convergence without sacrificing accuracy.

Cost Function. Fig. 2(c) shows that our proposed algorithm achieves the lowest cost, significantly outperforming interior-point and Newton's methods. Full and no offloading yield the highest costs, underscoring the limitations of static offloading in dynamic, resource-constrained scenarios.

3) Scenario 2: To evaluate our transform in heterogeneous networks, we summarized the parameter settings in Table II.

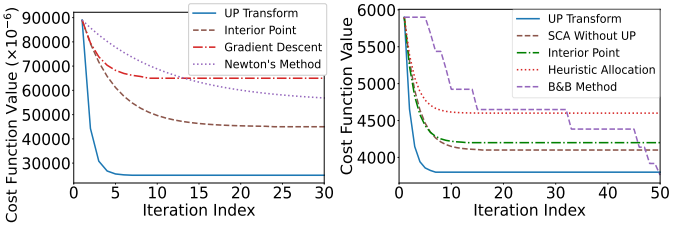
Implementation Details: In the heterogeneous network scenario, the primal optimization variables encompass the user association vectors $\boldsymbol{x}$, local computation frequencies $\boldsymbol{f}^l$, SBS computation allocations $\boldsymbol{f}^s$, transmission power $\boldsymbol{P}$, and computation offloading vectors $\boldsymbol{d}$. To ensure a valid starting point for the proposed SCA-based algorithm, we employ a feasible initialization strategy:

- The user association variables $x_{n,m}$ are relaxed to continuous values and randomly initialized in $[0, 1]$ satisfying $\sum_{m \in \mathcal{M}} x_{n,m} = 1$;
- Local frequencies f_n^l and transmission power P_n are randomly sampled from $[0, F_n]$ and $[0, P_n^{\max}]$, respectively;
- SBS computation resources f_n^s are initialized in $[0, F^s]$ and subsequently normalized to strictly satisfy the capacity constraint (22c).

Given these initial primal variables, the auxiliary variables $\boldsymbol{\mathcal{A}}$ defined in Theorem 4 are computed according to their closed-form expressions. Regarding the corresponding hyperparameters space $\boldsymbol{\mathcal{B}}$, we set the each element as 10^{-6}. Crucially, for the penalty method used to handle binary constraints, the penalty parameter is set to $\tau = 10^5$. The algorithm terminates when the relative objective change falls below $\epsilon = 10^{-4}$ or the maximum iteration count (set to 100) is reached.

Convergence Speed. As shown in Fig. 3(b), our proposed method converges in the fewest iterations, significantly outperforming all other baselines. This highlights the computational efficiency of UP transform, particularly in contrast to the B&B method, which suffers from high convergence latency due to its combinatorial search.

Cost Efficiency Evaluation. In terms of delay-energy tradeoff and cost (Fig. 3(b), 3(c)), the B&B algorithm achieves the global optimum but with much slower convergence. In contrast, our method reaches near-optimal solutions significantly faster, effectively balancing performance and efficiency—ideal for real-time optimization in heterogeneous edge computing.



(a) Partial Offloading Scenario (b) User Association Scenario

Fig. 4: Convergence Process of Proposed Scenarios

D. Convergence Trajectory Analysis

To further investigate the entire convergence process, Fig. 4 illustrates the explicit trajectories of the cost function, extending the iteration count analysis presented in Figs. 2(b) and 3(b). In the partial offloading scenario shown in Fig. 4(a), our proposed `UP` algorithm demonstrates a monotonic decrease and stabilizes at the minimum cost. In contrast, the baseline methods, such as Gradient Descent and Newton's Method, exhibit either oscillatory behavior or stagnation at local optima with significantly higher costs. For the user association scenario shown in Fig. 4(b), our proposed method is compared against B&B benchmark. The B&B curve shows a characteristic stepwise descent, resulting from discrete updates to the integer solutions over a prolonged search duration. Conversely, the proposed `UP` transform-based algorithm achieves a smooth convergence profile, reaching a near-optimal solution comparable to the B&B lower bound with substantially reduced computational overhead.

E. Hyperparameter Sensitivity Analysis

The sensitivity analysis is shown in Fig. 5 and reveals that c_n meets the properties introduced in *Remark* 5 (e.g., 10^{-8} to 10^{-6}), the final cost remains stable, indicating that the `UP` transform maintains high optimization accuracy across a wide range of small auxiliary parameters. However, as c_n increases beyond 10^{-4}, the final cost sharply increases, suggesting that an excessively large c_n hampers the optimization performance. Meanwhile, the number of convergence iterations remains nearly constant for all c_n values, demonstrating that the convergence speed is largely independent of the choice of c_n. This confirms that `UP` transform ensures stable and efficient convergence, with c_n mainly influencing the final solution quality rather than the optimization dynamics.

F. Scalability and Robustness

Cost Scalability with Varying User Loads N. Fig. 6 evaluates cost performance across different numbers of users. In both scenarios (partial offloading in Fig. 6(a) and user association in Fig. 6(b)), our algorithm consistently achieves the lowest costs, efficiently adapting offloading, associations, and resource allocations to dynamic system conditions.

Computation and Memory Constraints. Fig. 7 illustrates scalability under varying system constraints (server computing capacity F^e and task size C_n). In Fig. 7(a), our algorithm significantly outperforms baselines, with the lowest costs as F^e increases from 2GHz to 10GHz. Fig. 7(b) shows that, as task sizes grow ($[50, 100]$MB to $[300, 800]$MB), our algorithm maintains the slowest cost growth, highlighting its robustness in dynamically balancing local and edge resources under heavier workloads.

Cost Scalability with Varying SBSs M. Fig. 8 reports the average runtime of different algorithms as the number of SBSs M increases with a fixed number of users $N = 20$. Our proposed `UP` framework exhibits a mild and approximately linear growth in runtime with respect to M, which is consistent with the $\mathcal{O}(NM)$ scaling of the problem dimension and the absence of additional nested iterations investigated before in Section III-B. In contrast, generic continuous optimization methods, including IPM, Newton's methods, and gradient descent, incur noticeably higher computational costs as M increases due to their limited exploitation of problem structure. Moreover, the Branch-and-Bound (B&B) Method fails to converge within the prescribed time limit even for small network sizes, highlighting the prohibitive complexity of combinatorial optimization. These results demonstrate the favorable scalability of our proposed method for large-scale HetNet deployments.

G. `UpperBound` Transform versus `UP` Transform

Fig. 9 presents a comparative analysis of the convergence process of `UpperBound` transform (Theorem 1) and `UP` transform (Theorem 2) under two different initial settings of the optimization variables.

In Fig 9(a), we initialize the optimization variable $\boldsymbol{x}$ with strictly positive elements within the feasible domain $[0, 1]$. Though `UpperBound` transform performs comparably to the `UP` transform during the first few iterations, it meets convergence failure when $\boldsymbol{x}$ is optimized to 0 values. While in Fig. 9(b), some elements of $\boldsymbol{x}$ are initialized as zero. It is presented that the `UpperBound` transform fails to converge when the initial variable contains zero elements. This result is consistent with the theoretical analysis in Section II-B. In contrast, the `UP` transform successfully handles such scenarios by using hyperparameters to avoid undefined operations, ensuring stable updates of auxiliary variables throughout the iterative process.

These results empirically validate the robustness and general applicability of the `UP` transform in practical optimization problems, especially in settings with non-negative or mixed domain variables. The improved convergence behavior of the `UP` transform across varying initial conditions confirms its suitability for broader non-convex optimization tasks in edge computing and communication networks.

H. Additional Applicability Beyond MEC/HetNet

To further illustrate the general applicability of the proposed UP transform, we briefly discuss two representative wireless optimization scenarios beyond MEC and HetNet systems: wireless power control and energy-efficient federated learning (EE-FL). In both cases, the underlying optimization problems exhibit multiplicative or fractional coupling structures similar to the aforementioned scenarios considered, and can therefore be addressed within the same reformulation framework.

1) Wireless Power Control: A classic problem in multi-user interference networks is optimizing the transmit power vector

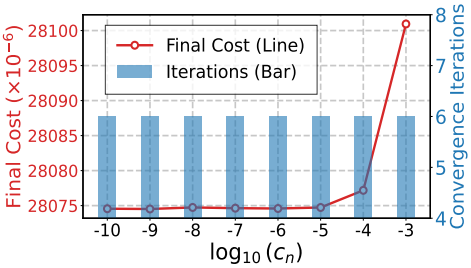
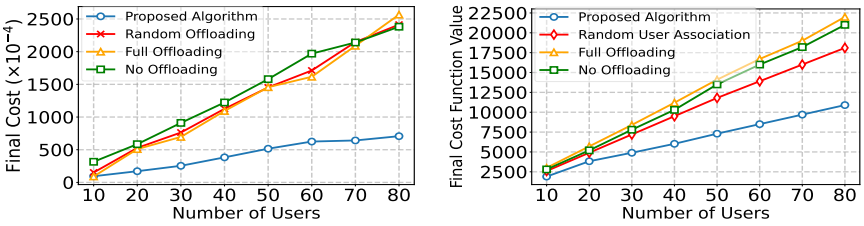
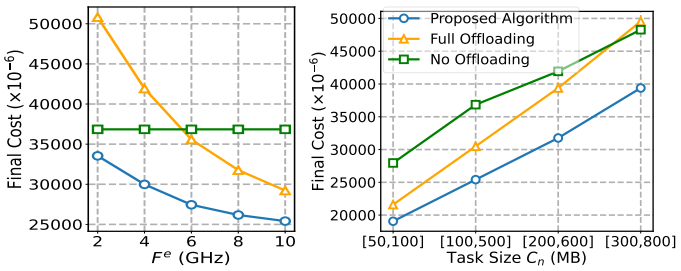


Fig. 5: Sensitivity Analysis to c_n



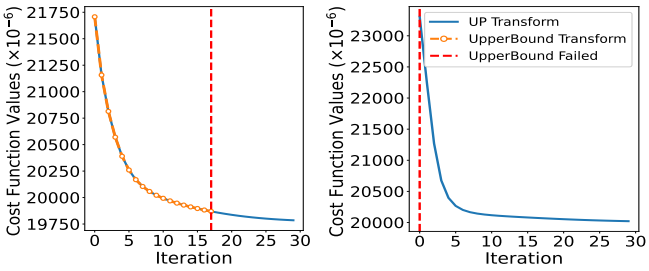
(a) Partial Offloading Scenario (b) User Association Scenario

Fig. 6: Cost function values vs. Number of users in proposed scenarios



(a) Impact of server computing (b) Impact of take size

Fig. 7: Scalability under Computation and Memory Constraints



(a) Initialized $\boldsymbol{x}$ without 0 Element (b) Initialized $\boldsymbol{x}$ with 0 Element

Fig. 9: UpperBound Transform vs UP Transform

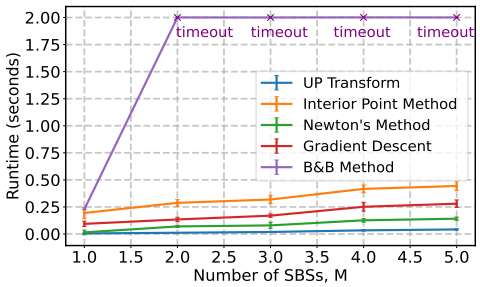


Fig. 8: Runtime Comparison versus the Number of SBSs M

$\boldsymbol{p} = \{p_k\}_{k\in\mathcal{K}}$ to maximize the weighted sum-rate, formulated as

$$\max_{\boldsymbol{p}\succeq\mathbf{0}} \sum_{k\in\mathcal{K}} w_k \log_2(1+\mathrm{SINR}_k).$$

The Signal-to-Interference-plus-Noise Ratio (SINR) for user k typically takes the fractional form:

$$\mathrm{SINR}_k(\boldsymbol{p}) = \frac{|h_{kk}|^2 p_k}{\sum_{j\neq k}|h_{kj}|^2 p_j + \sigma^2},$$

where the desired signal power (numerator) is coupled with the aggregate interference (denominator). Our proposed UP transform can effectively decouple the numerator and denominator by introducing auxiliary variables, converting the nonconvex sum-of-log-ratios problem into a sequence of tractable convex subproblems with respect to $\boldsymbol{p}$. Notably, in scenarios requiring discrete power levels (e.g., on/off states or quantized power control), the robustness of the UP transform against zero-valued denominators (as discussed in Remark 4) offers a distinct advantage over traditional fractional programming methods that may suffer from numerical instability when users become inactive.

2) Energy-Efficient Federated Learning (EE-FL): As another example, consider an EE-FL system, where multiple edge devices collaboratively train a global model. Let $\boldsymbol{f} = \{f_n\}_{n\in\mathcal{N}}$, $\boldsymbol{\tau} = \{\tau_n\}_{n\in\mathcal{N}}$, and $\boldsymbol{p} = \{p_n\}_{n\in\mathcal{N}}$ denote the local CPU frequency, transmission time, and transmit power of device n, respectively. A generic EE-FL optimization problem can be formulated as maximizing the sum of energy efficiencies:

$$\max_{\boldsymbol{f},\boldsymbol{\tau},\boldsymbol{p}} \sum_{n\in\mathcal{N}} \frac{Q_n(f_n,\tau_n,p_n)}{E_n^{\mathrm{comp}}(f_n)+E_n^{\mathrm{comm}}(\tau_n,p_n)}$$

where the numerator $Q_n(\cdot)$ represents the learning utility, and the denominator captures the total energy consumption, consisting of the computation energy $E_n^{\mathrm{comp}}(f_n)$ and communication energy $E_n^{\mathrm{comm}}(\tau_n,p_n)$.

The objective in (28) involves complex coupled terms in the denominator, particularly the summation of computation and communication energy, rendering the problem highly nonconvex. By leveraging the proposed UP transform, the coupled terms in the denominator (such as the product terms $E_n^{\mathrm{comm}}(\tau_n,p_n) = \tau_n p_n$) can be effectively decoupled via auxiliary variables, yielding a surrogate objective. This enables the use of alternating optimization or SCA-based methods to efficiently solve the resulting subproblems, while maintaining numerical stability even when some devices become inactive (e.g., zero frequency or power).

IV. Computation Complexity Analysis

In this section, we analyze computational complexity of our proposed transform and compare it with traditional algorithms.

A. Complexity of UP Transform

Before analyzing the complexity of the problem UP transform, the overview of the proposed algorithm adopting UP transform is presented in Algorithm 5. The KKT analysis is applied to update the optimization variable during each iteration. For the update phase of $\boldsymbol{y}^*$, step 8 requires $\mathcal{O}(N)$ per iteration. To evaluate the complexity of updating $\boldsymbol{x}^*$, we first assume the number of constraints in problem $\mathbb{P}_1$ is C and each $A_n(\boldsymbol{x})$ and $B_n(\boldsymbol{x})$ evaluation has complexity $\mathcal{O}(M)$ where M represents the dimension of $\boldsymbol{x}$. Then calculating the gradient $\nabla_{\boldsymbol{x}} W(\boldsymbol{x},\boldsymbol{y})$ has complexity $\mathcal{O}(N\cdot M)$ if $A_n(\boldsymbol{x})$ and $B_n(\boldsymbol{x})$ are linear or sparse forms. Calculating the gradients of constraints also contributes $\mathcal{O}(C\cdot M)$. Then, the total cost of gradient computation is $\mathcal{O}((N+C)\cdot M)$. Based on paper [32] and analysis of Appendix C and D, we approximate K as the

Algorithm 5: Algorithm of Adopting UP Transform

Input: Optimization problem $\mathbb{P}_1$, feasible set $\mathcal{X}$, optimization variable $\boldsymbol{x}$, and auxiliary variables $\boldsymbol{y}$;
Output: Optimal solutions $\boldsymbol{x}^*$ and auxiliary variables $\boldsymbol{y}^*$;
Randomly set the value of optimization variable $\boldsymbol{x}^{(0)}$ with $\boldsymbol{x}^{(0)} \in \mathcal{X}$;
Set the iteration number $j = 0$;
Initialize the current optimal solution $\boldsymbol{x}^* = \boldsymbol{x}^{(0)}$;
Initialize optimal auxiliary variables $\boldsymbol{y}^*$;
repeat
  Calculate values of the corresponding auxiliary variables $\boldsymbol{y}^{(j)}$ with $\boldsymbol{x}^*$ based on (10);
  Update optimal auxiliary variables $\boldsymbol{y}^* = \boldsymbol{y}^{(j)}$;
  Update the iteration number $j = j + 1$;
  Solve the problem $\mathbb{P}_2$ with given $\boldsymbol{y}^*$ by adopting KKT analysis;
  Obtain the current solution $\boldsymbol{x}^{(j)}$;
  Update the optimal solution $\boldsymbol{x}^* = \boldsymbol{x}^{(j)}$;
until *Convergence or Maximum iteration number* J;

running times, then the complexity of obtaining $\boldsymbol{x}^{(j)}$ through KKT analysis is $\mathcal{O}(K \cdot (M + C))$ as the KKT conditions involve a system of $M + C$ equations (from stationarity and primal feasibility) and $M+C$ unknowns (i.e., M optimization variable and C multipliers). Consequently, the worst complexity of Algorithm 5 is $\mathcal{O}(J \cdot ((N+C+K) \cdot M + K \cdot C + N))$, where J is the maximum iteration number.

B. Complexity Comparison

The computational complexities of our updating $\boldsymbol{x}^{(j)}$ algorithm with other traditional methods, consisting of Gradient Descent (GD) [33], Newton's Method [34], and the Interior-Point Method (IPM) [35], are comprehensively compared in Table III, where K_* is the maximum iterations required for update $x^{(j)}$ in optimization method $*$ and J_* denotes total outer iterations.

The **comparison** highlights the strengths of our proposed UP transform in solving optimization problems. Gradient Descent, with a per-iteration complexity of $\mathcal{O}((N + C) \cdot M)$, is simple and scales well for large dimensions M, but its slow convergence $\mathcal{O}(1/\epsilon^2)$ makes it inefficient for high-precision solutions. Newton's Method achieves faster quadratic convergence with a per-iteration complexity of $\mathcal{O}(M(N + C)(M + 1) + M^3)$, but the high cost of Hessian computation and solving linear systems limits its applicability to moderate-dimensional problems. The Interior-Point Method is highly effective for constrained optimization problems and is particularly effective for convex cases, converging logarithmically in $\mathcal{O}(1/\epsilon)$ iterations; however, its high per-iteration complexity $\mathcal{O}(M^3 + (N + C) \cdot M^2)$ makes it computationally expensive for large M. In contrast, our algorithm offers a balance of efficiency and scalability by directly solving optimality conditions with a per-iteration complexity of $\mathcal{O}(M+C)$. This approach avoids the computational burden of M^3-scaling seen in Newton's and Interior-Point methods, making it particularly advantageous for problems where feasible solutions can be efficiently identified. The strengths of our UP transform with KKT Analysis lie in its ability to directly address optimality conditions, its relatively lower dependence on dimensionality M, and its suitability for constrained problems where solving the KKT equations is computationally viable.

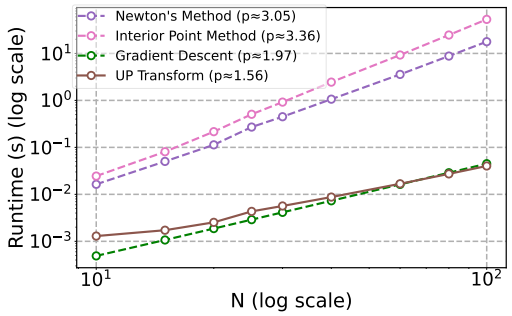


Fig. 10: Runtime Comparison of Optimization Methods

The theoretical complexity analysis is further supported by the **numerical runtime results** shown in Fig. 10. The runtime is reported in log-log scale as a function of the number of users N, where the slope (p) of each curve characterizes the growth rate. As observed, Newton's Method and Interior-Point Method exhibit near-cubic scaling behavior, which is consistent with their per-iteration computational cost dominated by $\mathcal{O}(M^3)$. Gradient descent shows an approximately quadratic growth trend, aligning with its $\mathcal{O}((N + C) \cdot M)$ per-iteration complexity under a nearly constant iteration count. In contrast, the proposed UP transform demonstrates a substantially lower runtime growth as N increases. Although the measured slope is slightly higher than the ideal linear scaling, this discrepancy can be attributed to implementation-level operations such as auxiliary variable updates and numerical search procedures, which are treated as constant-order terms in the theoretical analysis. Overall, the numerical results further confirm that the proposed UP transform scales more favorably than second-order and Interior-Point methods for increasing problem sizes.

C. Comparison with BCD and SCA Methods

We then discuss the conceptual positioning of the proposed UP transform relative to standard BCD and SCA frameworks. Rather than serving as a standalone optimization algorithm, UP transform functions as a reformulation technique that addresses nonconvex multiplicative or fractional coupling by introducing auxiliary variables and upper-bounding surrogates.

From a scalability perspective, UP transform reduces cross-variable coupling by enabling multiple decision variables to be optimized within a single block after transformation. This contrasts with conventional BCD approaches, where the number of optimization blocks typically grows with problem dimension, and with SCA-based methods, where scalability is often limited by the high computational cost of solving coupled convex subproblems (e.g., via Interior-Point methods).

In terms of computational characteristics, UP transform shifts the complexity associated with multiplicative coupling to convex subproblems that often admit low-complexity or closed-form solutions. As a result, the per-iteration complexity is significantly determined by the resulting convex subprograms, rather than by combinatorial nonconvex subproblems that may arise when standard BCD or SCA is applied.

TABLE III: Computation complexity comparison of different optimization methods

Algorithm	Per-iteration Complexity	Total Complexity
Our Algorithm	$\mathcal{O}(M+C)$	$\mathcal{O}(J((N+C+K)M+KC+N))$
Gradient Descent [33]	$\mathcal{O}((N+C)\cdot M)$	$\mathcal{O}(J_{GD}(K_{GD}(N+C)\cdot M+N))$
Newton's Method [34]	$\mathcal{O}(M(N+C)(M+1)+M^3)$	$\mathcal{O}(J_{NT}(K_{NT}(M(N+C)(M+1)+M^3)+N))$
Interior Point Method [35]	$\mathcal{O}(M(N+C)(M+1)+M^3)$	$\mathcal{O}(J_{IPM}K_{IPM}(M^3+(N+C)\cdot M^2))$

Regarding generalizability, UP transform is agnostic to the specific optimization framework and can be seamlessly integrated with BCD, SCA, or other iterative schemes. This modularity, combined with the inherent robustness against numerical singularities (as discussed in Remark 4), allows it to be applied to a broad class of problems involving continuous, discrete, or mixed decision variables, as illustrated by the considered application scenarios.

V. Comparison with Related Work

In this section, we compare the representative papers on FP and MP techniques in communication networks with our paper and emphasize the effectiveness of our proposed transforms.

A. Comparison with Related Studies

Comparing Our Theorem 1 with [2]. The result in the special case of convex $A_n(\boldsymbol{x})$, concave $C_n(\boldsymbol{x})$, convex $G(\boldsymbol{x})$, and convex set $\mathcal{X}$ has been presented in Section IV of work [2]. Theorem 1 in this paper does not impose the special requirements above and hence is useful for solving a wider set of optimization problems compared with [2]. Note that although we consider $\boldsymbol{x}^{(j)}$ as a KKT point of Problem $\mathbb{P}_2(\boldsymbol{y}^{(j)})$ in "②", this may still hold when $A_n(\boldsymbol{x})$ is not convex, $C_n(\boldsymbol{x})$ is not concave (resp., $B_n(\boldsymbol{x})$ is not convex), $G(\boldsymbol{x})$ is not convex, or set $\mathcal{X}$ is not convex, because KKT conditions may still hold for certain non-convex optimization problems [36].

Comparing Our Theorem 2 with [2]. From the above, we present Theorem 1 as a general transform form of the minimization optimization problem and always holds when the introduced auxiliary variable is not zero (i.e., $\boldsymbol{y} \neq \mathbf{0}$). Therefore, the comparison between Theorem 2 and [2] is inherently a comparison between Theorem 1 and Theorem 2. With the introduction of Theorem 2, it can be adapted to the more general minimization problem, i.e., Theorem 2 can be adapted to the case of $\boldsymbol{y} = \mathbf{0}$ while also being applicable to the cases mentioned by Theorem 1.

Comparison with [37], [38]. Not like the UpperBound transform, the *quadratic transform* proposed in [37], [38] can't be applied in the minimization case when $A_n(\boldsymbol{x})$ is convex and $B_n(\boldsymbol{x})$ is concave in Theorem 1. The minimization, in this case, would be $-\infty$ when using the *quadratic transform*. Besides, some minor issues arise from "*Corollary 1*" in [37] regarding its accuracy. This corollary posits an equivalence between the solutions of the original sum-of-ratios problem and those derived via the *quadratic transform*. However, this assertion is flawed because the properties of the solutions found through the *quadratic transform* differ significantly from those of the original problem. The *quadratic transform* primarily identifies a KKT point, which does not necessarily coincide with the optimal point of the original sum-of-ratios problem. Therefore, declaring these two problems as equivalent is incorrect.

TABLE IV: Overview of representative papers adopting the AO method to find the optimal solutions.

Paper	Optimization variables which are optimized alternatively
[40]	User association $\boldsymbol{A}$; Computation capacity, power, and 3D location planning $\{\boldsymbol{F}, \boldsymbol{P}, \boldsymbol{Z}\}$
[41]	UAV scheduling and association$\{\boldsymbol{A}, \boldsymbol{Q}\}$; Transmitting power $\boldsymbol{P}$
[42]	User association $\boldsymbol{\mu}$; Transmission power allocation $\boldsymbol{p}$; Location of UAV $\boldsymbol{f}$
[43]	GUE association strategy $\boldsymbol{A}$; Transmit precoding matrix $\{\mathbb{Q}, \mathbb{W}\}$ of GUE and HAP
[44]	UBS resource allocation β; UBS placement $\boldsymbol{x^{ubs}}$, and user association $\boldsymbol{z}$
[45]	E2E association $\boldsymbol{x}$; Allocation of uplink and downlink transmit bandwidth, computation, power, and time duration $\{\bar{\boldsymbol{b}}, \underline{\boldsymbol{b}}, \tilde{\boldsymbol{f}}, \underline{\boldsymbol{p}}, \boldsymbol{\tau}\}$

Comparison with [39]. The authors in [39] proposed a methodology for identifying the global optimum solution to the sum-of-ratios problem under the condition that $G(x) = 0$ in Theorem 1. But this method fails in cases where $G(x) \neq 0$. In contrast, our theorems still hold when $G(x) \neq 0$.

B. Other Related Work

We introduce representative studies adopting the alternating optimization (AO) method in Table IV. In scenarios concerning unmanned aerial vehicles (UAVs) [40]–[42], ground user equipment (GUE) and high altitude platforms (HAPs) [43], UAV-mounted base stations (UBSs) [44], and end-to-end (E2E) devices [45], problems are rarely solved by transforming the original objective. Instead, they are typically addressed by iteratively fixing subsets of variables to handle constraints. While this approach yields solutions, it significantly increases computational complexity, often requiring distinct algorithms (e.g., brute-force search or gradient ascent) for each subproblem. In contrast, our UP and UpperBound transforms offer a unified approach to jointly optimize all variables, reducing computational overhead and improving efficiency.

VI. Conclusion

In this paper, we introduced the generalized UP transform to overcome the limitations of the original UpperBound transform in solving complex non-convex optimization problems characterized by multiple ratio structures. Through rigorous theoretical analysis, we established that our proposed UP transform reliably converges to a KKT point. Extensive simulations validated its superior performance, demonstrating significant improvements in convergence speed and solution quality in practical communication networks.

Appendix A
Proof of Theorem 1

Since $(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})$ converges to $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ as $j \to \infty$, we obtain from (7) that

$$y_n^* = \frac{B_n(\boldsymbol{x}^*)}{2A_n(\boldsymbol{x}^*)}. \tag{28}$$

Since $\boldsymbol{x}^{(j)}$ is a KKT point of $\mathbb{P}_2(\boldsymbol{y}^{(j)})$, and $(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})$ converges to $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ as $j \to \infty$, we know that $\boldsymbol{x}^*$ is a KKT point of $\mathbb{P}_2(\boldsymbol{y}^*)$. Then supposing that $\boldsymbol{x} \in \mathcal{X}$ means

$$\begin{cases} \mathcal{Q}_q(\boldsymbol{x}) \leq 0, & \text{for } q = 1, 2, \ldots, Q, \\ \mathcal{R}_r(\boldsymbol{x}) = 0, & \text{for } r = 1, 2, \ldots, R, \\ \boldsymbol{x} \in \mathbb{R}^M, \end{cases} \tag{29}$$

we have the following set of KKT conditions, where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ denote the Lagrange multipliers:

$$\begin{cases} \text{Stationarity:} \\ \Big[\frac{\partial}{\partial x_m}\big(W(\boldsymbol{x}, \boldsymbol{y}^*) + \sum_{q=1}^{Q} \alpha_q \mathcal{Q}_q(\boldsymbol{x}) \\ + \sum_{r=1}^{R} \beta_r \mathcal{R}_r(\boldsymbol{x})\big)\Big]|_{\boldsymbol{x}=\boldsymbol{x}^*} = 0, \text{for } m = 1, \ldots, M, & \text{(30a)} \\ \text{Primal feasibility:} \\ \mathcal{Q}_q(\boldsymbol{x}^*) \leq 0, \text{ for } q = 1, 2, \ldots, Q, & \text{(30b)} \\ \mathcal{R}_r(\boldsymbol{x}^*) = 0, \text{ for } r = 1, 2, \ldots, R, & \text{(30c)} \\ \text{Dual feasibility:} \\ \alpha_q \geq 0 \text{ for } q = 1, 2, \ldots, Q, & \text{(30d)} \\ \text{Complementary slackness:} \\ \alpha_q \mathcal{Q}_q(\boldsymbol{x}^*) = 0 \text{ for } q = 1, 2, \ldots, Q. & \text{(30e)} \end{cases}$$

Recalling the expressions of $W(\boldsymbol{x}, \boldsymbol{y})$ in Equations (5) and (2) and $H(\boldsymbol{x})$ in (3), we can leverage (28) to show the equivalence between $\frac{\partial}{\partial x_m} W(\boldsymbol{x}, \boldsymbol{y}^*)|_{\boldsymbol{x}=\boldsymbol{x}^*}$ and $\frac{\partial}{\partial x_m} H(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}^*}$ so that (30a) is equivalent to

$$\Big[\frac{\partial}{\partial x_m}\big(H(\boldsymbol{x}) + \sum_{q=1}^{Q} \alpha_q \mathcal{Q}_q(\boldsymbol{x}) + \sum_{r=1}^{R} \beta_r \mathcal{R}_r(\boldsymbol{x})\big)\Big]\Big|_{\boldsymbol{x}=\boldsymbol{x}^*} = 0,$$

$$\text{for } m = 1, 2, \ldots, M. \tag{31}$$

Clearly, (31) (30b) (30c) (30d) (30e) together mean that $\boldsymbol{x}^*$ is a KKT point for Problem $\mathbb{P}_1$ in (3). □

Appendix B
Proof of Theorem 2

To prove Theorem 2, we first prove that

Lemma 1. *$(\boldsymbol{y}^{(j)}, \boldsymbol{x}^{(j)})|_{j=1,2,\ldots}$ in Theorem 2 are solving Problem $\mathbb{P}_2'$ defined below via alternating optimization (AO):*

$$\mathbb{P}_2'\text{: } \underset{\boldsymbol{x},\boldsymbol{y}}{\textit{minimize}} \;\; W(\boldsymbol{x}, \boldsymbol{y})$$
$$\textit{subject to} \;\; \begin{matrix} \boldsymbol{x} \in \mathcal{X} \textit{ and} \\ y_n \geq c_n, \; \forall n \in \mathcal{N}, \end{matrix}$$

where the expression of $W(\boldsymbol{x}, \boldsymbol{y})$ is given via Equations (2) (5).

With $\mathbb{P}_2'(\boldsymbol{x})$ (resp., $\mathbb{P}_2'(\boldsymbol{y})$) denoting the problem of optimizing $\boldsymbol{y}$ given $\boldsymbol{x}$ (resp., optimizing $\boldsymbol{x}$ given $\boldsymbol{y}$) for $\mathbb{P}_2'$, it is straightforward to see that $\boldsymbol{y}^{(j)}$ given by (10) optimally solves $\mathbb{P}_2(\boldsymbol{x}^{(j-1)})$). In addition, $\mathbb{P}_2'(\boldsymbol{y})$ is the same as $\mathbb{P}_2(\boldsymbol{y})$ in (9), so $\boldsymbol{x}^{(j)}$ as a KKT point of $\mathbb{P}_2(\boldsymbol{y}^{(j)})$ is also a KKT point of $\mathbb{P}_2'(\boldsymbol{y}^{(j)})$. Based on the above, Lemma 1 is proved. Then we have the convergence of $(\boldsymbol{y}^{(j)}, \boldsymbol{x}^{(j)})|_{j=1,2,\ldots}$ due to the AO process of solving $\mathbb{P}_2'$.

With $(\boldsymbol{x}^{\#}, \boldsymbol{y}^{\#})$ denoting the convergence of $(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})$ as $j \to \infty$, we know from (10) that $y_n^{\#}$ being the n-th dimension of $\boldsymbol{y}^{\#}$ becomes

$$y_n^{\#} = \max\left\{\frac{B_n(\boldsymbol{x}^{\#})}{2A_n(\boldsymbol{x}^{\#})}, \; c_n\right\}, \; \forall n \in \mathcal{N}. \tag{32}$$

Readers may observe the difference between (32) above and (28) in the proof of Theorem 1. Then, as elaborated below, the remaining steps to complete showing Theorem 2 are similar to those after (28) in Theorem 1's proof.

Since $\boldsymbol{x}^{(j)}$ is a KKT point of $\mathbb{P}_2'(\boldsymbol{y}^{(j)})$, and $(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})$ converges to $(\boldsymbol{x}^{\#}, \boldsymbol{y}^{\#})$ as $j \to \infty$, we know that $\boldsymbol{x}^{\#}$ is a KKT point of $\mathbb{P}_2'(\boldsymbol{y}^{\#})$. Then supposing that $\boldsymbol{x} \in \mathcal{X}$ means (29), we have the following set of KKT conditions, where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ denote the Lagrange multipliers:

$$\begin{cases} \text{Stationarity:} \\ \Big[\frac{\partial}{\partial x_m}\big(W(\boldsymbol{x}, \boldsymbol{y}^{\#}) + \sum_{q=1}^{Q} \alpha_q \mathcal{Q}_q(\boldsymbol{x}) \\ + \sum_{r=1}^{R} \beta_r \mathcal{R}_r(\boldsymbol{x})\big)\Big]|_{\boldsymbol{x}=\boldsymbol{x}^{\#}} = 0, \text{for } m = 1, \ldots, M, & \text{(33a)} \\ \text{Primal feasibility:} \\ \mathcal{Q}_q(\boldsymbol{x}^{\#}) \leq 0, \text{ for } q = 1, 2, \ldots, Q, & \text{(33b)} \\ \mathcal{R}_r(\boldsymbol{x}^{\#}) = 0, \text{ for } r = 1, 2, \ldots, R, & \text{(33c)} \\ \text{Dual feasibility:} \\ \alpha_q \geq 0 \text{ for } q = 1, 2, \ldots, Q, & \text{(33d)} \\ \text{Complementary slackness:} \\ \alpha_q \mathcal{Q}_q(\boldsymbol{x}^{\#}) = 0 \text{ for } q = 1, 2, \ldots, Q. & \text{(33e)} \end{cases}$$

Recalling the expressions of $W(\boldsymbol{x}, \boldsymbol{y})$ in Equations (5) (2) and $L_{\boldsymbol{c}}(\boldsymbol{x})$ in (11), we can use (32) to show the equivalence between $\frac{\partial}{\partial x_m} W(\boldsymbol{x}, \boldsymbol{y}^{\#})|_{\boldsymbol{x}=\boldsymbol{x}^{\#}}$ and $\frac{\partial}{\partial x_m} L_{\boldsymbol{c}}(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}^{\#}}$ so that (33a) is equivalent to

$$\Big[\frac{\partial}{\partial x_m}\big(L_{\boldsymbol{c}}(\boldsymbol{x}) + \sum_{q=1}^{Q} \alpha_q \mathcal{Q}_q(\boldsymbol{x}) + \sum_{r=1}^{R} \beta_r \mathcal{R}_r(\boldsymbol{x})\big)\Big]\Big|_{\boldsymbol{x}=\boldsymbol{x}^{\#}} = 0,$$

$$\text{for } m = 1, 2, \ldots, M. \tag{34}$$

Clearly, (33b) (33c) (33d) (33e) (34) together mean that $\boldsymbol{x}^{\#}$ is a KKT point for Problem $\mathbb{P}_1'$ in (11). □

Appendix C
Proof of Theorem 3

Deriving from the functional expression $D_n(x_n)$ as delineated in (18a), two properties of this condition can be inferred contingent upon the values of auxiliary variables and Lagrange multipliers: *1). $D_n(x_n)$ is non-decreasing for x_n and 2). Specifically, we can obtain the explicit expression $\widetilde{x}_n = D_n^{-1}(0)$ by setting $\beta = 0$ and $\gamma_n = 0$ if $D_n(x_n) = 0$ has a solution.* Then we proceed to discuss the different cases based on conditions (19a) and (19b) which are outlined below:

- Case 1: $\widetilde{x}_n \geq 1$. In this case, we can infer that $D_n(x_n)$ is equal to or less than zero (i.e., $D_n(1) \leq 0$). Therefore, the optimal solution can be set that $\widetilde{x}_n = 1$ and $\beta_n = 0$ to meet the conditions, and then the value of γ_n is equal to $-D_n(1)$ which exactly meets the condition (19b).
- Case 2: $0 < \widetilde{x}_n < 1$. We can simply set the optimal value of optimization variable x_n as $\widetilde{x}_n$ with $\beta_n = 0$ and $\gamma_n = 0$ based on the second property.
- Case 3: $\widetilde{x}_n \leq 0$. Conversely with Case 1, $\widetilde{x}_n \leq 0$ means when $x_n = 0$, $D_n(0)$ is equal to or better than 0. We choose the feasible solution as $\widetilde{x}_n = 0$,$\gamma_n = 0$, and $\beta_n = D_n(0)$.

Thus, we summarize all of the above cases as follows:

$$\begin{cases} x_n^* = \max\{\min\{\widetilde{x}_n, 1\}, 0\}; \\ \beta_n^* = \max\{D_n(0), 0\}; \\ \gamma_n^* = -\min\{D_n(1), 0\}; \end{cases} \tag{35}$$

Based on the convexity of the function $H_{n,1}(f_n^l)$, it can be inferred that the term $[H_{n,1}(f_n^l)]^2(u_n^{(k)} + c_n^{(k)})$ maintains convexity with the scalar composition theory (Page 84 in [46]).

We next obtain the *non-decreasing nature* of the function $Q_n(f_n^l)$ as the decoupling of each optimization variable and being obtained by first-order derivation of the mentioned term. Then we can discuss the similar cases with x_n by assuming $\widetilde{f}_n^l = Q_n^{-1}(0)$ and summarize the discussion as follows:

$$\begin{cases} {f_n^l}^* = \max\{\min\{\widetilde{f}_n^l, F_n^l\}, 0\}; \\ \epsilon_n^* = \max\{Q_n(0), 0\}; \\ \zeta_n^* = -\min\{Q_n(F_n^l), 0\}; \end{cases} \tag{36}$$

According to the analysis of the optimization variable f_n^l, we obtain similar characteristics of the variable f_n^e in the function $R_n(f_n^e, \delta)$. Assume that when $R_n(f_n^e, \delta) = 0$, the value expression of f_n^e is denoted as $\widehat{f}_n^e(\delta)$. We discuss the cases based on the (18c):

- Case 1: $\widehat{f}_n^e(\delta) \geq F_n^e$. In this case, we can infer that $R_n(F_n^e, \delta) \leq 0$). Therefore, the optimal solution can be set that $\widehat{f}_n^e(\delta) = F_n^e$ and $\eta_n = 0$ to meet the conditions, and then the value of θ_n is $-R_n(F_n^e, \delta)$ which exactly meets the conditions.
- Case 2: $0 < \widehat{f}_n^e(\delta) < 1$. We can set the optimal value as $\widehat{f}_n^e(\delta)$ with $\eta_n = 0$ and $\theta_n = 0$ based on the second property.
- Case 3: $\widehat{f}_n^e(\delta) \leq 0$. Unlike Case 1, $\widehat{f}_n^e(\delta) \leq 0$ means when $\widehat{f}_n^e(\delta) = 0$, $R_n(0, \delta)$ is equal to or better than 0. Therefore, we choose the feasible solution as $\widehat{f}_n^e(\delta) = 0$, $\theta_n = 0$, and $\eta_n = R_n(0, \delta)$.

Therefore, for the above analysis, we summarize that:

$$\begin{cases} \widetilde{f}_n^e(\delta) = \max\{\min\{\widehat{f}_n^e(\delta), F_n^e\}, 0\}; \\ \widetilde{\eta}_n(\delta) = \max\{R_n(0, \delta), 0\}; \\ \widetilde{\theta}_n(\delta) = -\min\{R_n(F_n^e, \delta), 0\}; \end{cases} \tag{37}$$

For the optimization variable f_n^e, we cannot directly obtain the feasible solution while getting the formulation related to the multiplier δ. Drawing from the condition (19c), we discuss the value of δ based on the scenarios assuming $\delta = 0$:

- Case 1: $\sum_{n\in\mathcal{N}} \widetilde{f}_n^e(0) \leq F^e$. In this situation, we set $\delta = 0$ to meet the condition (20c).
- Case 2: $\sum_{n\in\mathcal{N}} \widetilde{f}_n^e(0) > F^e$. Conversely, we need to find the optimal value denoted as $\widetilde{\delta}$ of the δ based on the condition (19c) by adopting the bisection method.

Consequently, the feasible solution of δ is expressed as:

$$\delta^* = \begin{cases} 0, & \sum_{n\in\mathcal{N}} \widetilde{f}_n^e(0) \leq F^e, \\ \widetilde{\delta}, & \text{others.} \end{cases} \tag{38}$$

Until now, the optimal solution of the optimization variable $[\boldsymbol{x}^*, {\boldsymbol{f}^{\boldsymbol{l}}}^*, {\boldsymbol{f}^{\boldsymbol{e}}}^*]$ and Lagrange multiplier $[\boldsymbol{\beta}^*, \boldsymbol{\gamma}^*, \delta^*, \boldsymbol{\epsilon}^*, \boldsymbol{\zeta}^*, \boldsymbol{\eta}^*, \boldsymbol{\theta}^*]$ can be obtained by above analysis. □

APPENDIX D
PROOF OF THEOREM 4

To decouple the optimization variable in each term of the objective function, we adopt the UP transform and introduce auxiliary variables in the $(k+1)$-th iteration as follows:

For $O_n(\boldsymbol{x}, \boldsymbol{f^l}, \boldsymbol{d})$, we introduce the auxiliary variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Then the function in the $(k+1)$-th iteration can be transformed into $\widetilde{O}_n(\boldsymbol{x}, \boldsymbol{f^l}, \boldsymbol{d} \mid \boldsymbol{\alpha}^{(k)}, \boldsymbol{\beta}^{(k)}, \boldsymbol{b}^{(k)}, \boldsymbol{e}^{(k)})$ which equals

$$\sum_{m\in\mathcal{M}} [\widehat{O}_{n,m}(\boldsymbol{f^l}, \boldsymbol{d} | \boldsymbol{\beta}^{(k)}, \boldsymbol{e}^{(k)})]^2 (\alpha_{n,m}^{(k)} + b_{n,m}^{(k)}) + \frac{x_{n,m}^2 c_n}{4(\alpha_{n,m}^{(k)} + b_{n,m}^{(k)})},$$

where $\widehat{O}_{n,m}(\boldsymbol{f^l}, \boldsymbol{d} | \boldsymbol{\beta}^{(k)}, \boldsymbol{e}^{(k)}) = (\frac{w_1}{f_n^l} + w_2 k^l {f_n^l}^2)^2 (\beta_{n,m}^{(k)} + e_{n,m}^{(k)}) + \frac{(d_n - d_{n,m})^2}{4(\beta_{n,m}^{(k)} + e_{n,m}^{(k)})}$. When $x_{n,m} \neq 0$ and $d_n - d_{n,m} \neq 0$, values of auxiliary variables are $\beta_{n,m}^{(k)} = \frac{d_n - d_{n,m}^{(k)}}{2[\frac{w_1}{{f_n^l}^{(k)}} + w_2 k^l ({f_n^l}^{(k)})^2]}$ and $\alpha_{n,m}^{(k)} = \frac{x_{n,m}^{(k)} c_n}{2\widehat{O}_{n,m}(\boldsymbol{f^l}^{(k)}, \boldsymbol{d}^{(k)} \mid \boldsymbol{\beta}^{(k)}, \boldsymbol{e}^{(k)})}$ and the selection of constant $\boldsymbol{b}^{(k)}$ and $\boldsymbol{e}^{(k)}$ follows the properties in Remark 5 and the following constants obey the same rules.

For function $S_n(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{d})$, it can be transformed as $\widetilde{S}_n(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{d} \mid \boldsymbol{\gamma}^{(k)}, \boldsymbol{\delta}^{(k)}, \boldsymbol{g}^{(k)}, \boldsymbol{h}^{(k)})$ which equals:

$$\sum_{m\in\mathcal{M}} [\widehat{S}_{n,m}(\boldsymbol{P}, \boldsymbol{d} \mid \boldsymbol{\delta}^{(k)}, \boldsymbol{h}^{(k)})]^2 (\gamma_{n,m}^{(k)} + g_{n,m}^{(k)}) + \frac{x_{n,m}^2}{4(\gamma_{n,m}^{(k)} + g_{n,m}^{(k)})},$$

where $\widehat{S}_{n,m}(\boldsymbol{P}, \boldsymbol{d} | \boldsymbol{\delta}^{(k)}, \boldsymbol{h}^{(k)}) = [\overline{S}(P_n)]^2 (\delta_{n,m}^{(k)} + h_{n,m}^{(k)}) + \frac{d_{n,m}^2}{4(\delta_{n,m}^{(k)} + h_{n,m}^{(k)})}$ and $\overline{S}(P_n) = \frac{w_1 + w_2 P_n}{B_m \ln(1 + \frac{P_n H_{n,m}}{\sigma^2})}$. $\boldsymbol{g}^{(k)}, \boldsymbol{h}^{(k)}$ are expressed as:

$$\delta_{n,m}^{(k)} = \frac{d_{n,m}^{(k)}}{2\overline{S}(P_n^{(k)})}, \gamma_{n,m}^{(k)} = \frac{x_{n,m}^{(k)}}{2\widehat{S}_{n,m}(\boldsymbol{P}^{(k)}, \boldsymbol{d}^{(k)} \mid \boldsymbol{\delta}^{(k)}, \boldsymbol{h}^{(k)})},$$

when the constants are equal to zero.

For function $U_n(\boldsymbol{x}, \boldsymbol{f^s}, \boldsymbol{d})$, we introduce the auxiliary $\boldsymbol{\epsilon}$ and $\boldsymbol{\zeta}$ to transform it to $\widetilde{U}_n(\boldsymbol{x}, \boldsymbol{f^s}, \boldsymbol{d} \mid \boldsymbol{\epsilon}^{(k)}, \boldsymbol{\zeta}^{(k)}, \boldsymbol{j}^{(k)}, \boldsymbol{l}^{(k)})$ being equal to $[\widehat{U}_n(\boldsymbol{f^s}, \boldsymbol{d} \mid \boldsymbol{\zeta}^{(k)}, \boldsymbol{l}^{(k)})]^2 (\epsilon_n^{(k)} + j_n^{(k)}) + \frac{x_{n,1}^2}{4(\epsilon_n^{(k)} + j_n^{(k)})}$, where the constants are denoted as $\boldsymbol{j}^{(k)}$ and $\boldsymbol{l}^{(k)}$. Function $\widehat{U}_n$ is expressed as:

$$\widehat{U}_n(\boldsymbol{f^s}, \boldsymbol{d} | \boldsymbol{\zeta}^{(k)}, \boldsymbol{l}^{(k)}) = (\frac{w_1}{f_n^s} + w_2 k^S {f_n^s}^2)^2 (\zeta_n^{(k)} + l_n^{(k)}) + \frac{(d_{n,1} - d_{n,1}^{'})^2}{4(\zeta_n^{(k)} + l_n^{(k)})}.$$

Based on Equation (10), if constants are equal to 0, auxiliary variables can be inferred as:

$$\zeta_n^{(k)} = \frac{d_{n,1}^{(k)} - d_{n,1}^{'(k)}}{2\left[\frac{w_1}{f_n^{s(k)}} + w_2 k^S (f_n^{s(k)})^2\right]}, \epsilon_n^{(k)} = \frac{x_{n,1}^{(k)}}{2\widehat{U}_n(\boldsymbol{f^s}^{(k)}, \boldsymbol{d}^{(k)} \mid \boldsymbol{\zeta}^{(k)}, \boldsymbol{l}^{(k)})}.$$

For the last function $V_n(\boldsymbol{x}, \boldsymbol{d})$, it can be similarly transformed to $\widetilde{V}_n(\boldsymbol{x}, \boldsymbol{d} \mid \boldsymbol{\eta}^{(k)}, \boldsymbol{\theta}^{(k)}, \boldsymbol{q}^{(k)}, \boldsymbol{z}^{(k)})$, where $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$ are auxiliary variables, and $\boldsymbol{q}$ and $\boldsymbol{z}$ are the constants. Then the details of $\widetilde{V}_n$ is: $\widetilde{V}_n = [d_{n,1}^{'2} (\eta_n^{(k)} + q_n^{(k)}) + \frac{x_{n,1}^2}{4(\eta_n^{(k)} + q_n^{(k)})}] C + [d_{n,2}^2 (\theta_n^{(k)} + z_n^{(k)}) + \frac{x_{n,2}^2}{4(\theta_n^{(k)} + z_n^{(k)})}] (C - \frac{w_1 + w_2 \bar{P}}{r_0})$. When auxiliary variables are not equal to zero, auxiliary variables are expressed as $\eta_n^{(k)} = \frac{x_n^{(k)}}{2d_{n,1}^{'(k)}}$ and $\theta_n^{(k)} = \frac{x_{n,2}^{(k)}}{2d_{n,2}^{(k)}}$.

Until now, the optimization variable in the term $w_1 T_n + w_2 E_n$ of the objective function in the Equation (25) is transformed into a convex function by using the UP transform. □

REFERENCES

[1] Z. Wang, Y. Wang, and J. Zhao, "Resource Allocation and Secure Wireless Communication in the Large Model based Mobile Edge Computing System," in *MobiHoc '24*, 2024, p. 111–120. [Online]. Available: https://doi.org/10.1145/3641512.3686373

[2] J. Zhao, L. Qian, and W. Yu, "Human-Centric Resource Allocation in the Metaverse Over Wireless Communications," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 42, no. 3, pp. 514–537, 2024.

[3] K. Shen and W. Yu, "Fractional Programming for Communication Systems—Part I: Power Control and Beamforming," *IEEE Transactions on Signal Processing*, vol. 66, no. 10, pp. 2616–2630, 2018.

[4] A. Seetharam, B. Jiang, D. Goeckel, J. Kurose, and R. Hancock, "Optimizing Control Overhead for Power-Aware Routing in Wireless Networks," in *MILCOM 2013-2013 IEEE Military Communications Conference*. IEEE, 2013, pp. 870–875.

[5] H. Konno and T. Kuno, "Multiplicative Programming Problems," *Handbook of Global Optimization*, pp. 369–405, 1995.

[6] D. Malak, F. V. Mutlu, J. Zhang, and E. M. Yeh, "Joint power control and caching for transmission delay minimization in wireless hetnets," *IEEE/ACM Transactions on Networking*, vol. 32, no. 2, pp. 1477–1492, 2024.

[7] Y. Deng, F. Lyu, T. Xia, Y. Zhou, Y. Zhang, J. Ren, and Y. Yang, "A communication-efficient hierarchical federated learning framework via shaping data distribution at edge," *IEEE/ACM Transactions on Networking*, vol. 32, no. 3, pp. 2600–2615, 2024.

[8] R. Liu, K. Guo, K. An, F. Zhou, Y. Wu, Y. Huang, and G. Zheng, "Resource allocation for noma-enabled cognitive satellite–uav–terrestrial networks with imperfect csi," *IEEE Transactions on Cognitive Communications and Networking*, vol. 9, no. 4, pp. 963–976, 2023.

[9] J. Charles Gilbert, C. C. Gonzaga, and E. Karas, "Examples of ill-behaved central paths in convex optimization," *Mathematical programming*, vol. 103, no. 1, pp. 63–94, 2005.

[10] S. Schaible and T. Ibaraki, "Fractional Programming," *European journal of operational research*, vol. 12, no. 4, pp. 325–338, 1983.

[11] J. C. Bezdek and R. J. Hathaway, "Some Notes on Alternating Optimization," in *Advances in Soft Computing—AFSS 2002: 2002 AFSS International Conference on Fuzzy Systems Calcutta, India, February 3–6, 2002 Proceedings*. Springer, 2002, pp. 288–300.

[12] M. Dhar, A. Debnath, B. K. Bhattacharyya, M. K. Debbarma, and S. Debbarma, "A comprehensive study of different objectives and solutions of controller placement problem in software-defined networks," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 5, p. e4440, 2022.

[13] P. Peng, W. Wu, W. Lin, F. Zhang, Y. Liu, and K. Li, "Reliable task offloading in sustainable edge computing with imperfect channel state information," *IEEE Transactions on Network and Service Management*, vol. 21, no. 6, pp. 6423–6436, 2024.

[14] Z. Fu, J. Liu, Y. Mao, L. Qu, L. Xie, and X. Wang, "Energy-efficient uav-assisted federated learning: Trajectory optimization, device scheduling, and resource management," *IEEE Transactions on Network and Service Management*, vol. 22, no. 2, pp. 974–988, 2025.

[15] M. Asim, M. ELAffendi, and A. A. A. El-Latif, "Multi-IRS and Multi-UAV-assisted MEC system for 5G/6G networks: Efficient joint trajectory optimization and passive beamforming framework," *IEEE Transactions on Intelligent Transportation Systems*, pp. 4553–4564, Apr. 2023.

[16] S. Zawad, X. Ma, J. Yi, C. Li, M. Zhang, L. Yang, F. Yan, and Y. He, "FedCust: Offloading Hyperparameter Customization for Federated Learning," *Performance Evaluation*, vol. 167, p. 102450, 2025.

[17] X. Qin, Q. Xie, and B. Li, "Distributed Threshold-Based Offloading for Heterogeneous Mobile Edge Computing," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2023, pp. 202–213.

[18] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, Opportunities, and Directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[19] Y. Huang, T. Sun, and T. He, "Overlay-based decentralized federated learning in bandwidth-limited networks," in *Proceedings of the Twenty-fifth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, 2024, pp. 121–130.

[20] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, 2018.

[21] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency Minimization for D2D-Enabled Partial Computation Offloading in Mobile Edge Computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4472–4486, 2020.

[22] Y. Zhang, Z. Kuang, Y. Feng, and F. Hou, "Task offloading and trajectory optimization for secure communications in dynamic user multi-uav mec systems," *IEEE Transactions on Mobile Computing*, 2024.

[23] Y. Cong, K. Xue, C. Wang, W. Sun, S. Sun, and F. Hu, "Latency-energy joint optimization for task offloading and resource allocation in mec-assisted vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 12, pp. 16 369–16 381, 2023.

[24] C. Dong, Y. Tian, Z. Zhou, W. Wen, and X. Chen, "Joint power allocation and task offloading for reliability-aware services in noma-enabled mec," *IEEE Transactions on Wireless Communications*, vol. 23, no. 7, pp. 7537–7551, 2024.

[25] L. Tan, Z. Kuang, L. Zhao, and A. Liu, "Energy-efficient joint task offloading and resource allocation in ofdma-based collaborative edge computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 3, pp. 1960–1972, 2022.

[26] H. Chen, H. Cui, J. Wang, P. Cao, Y. He, and M. Guizani, "Computation offloading optimization for uav-based cloud-edge collaborative task scheduling strategy," *IEEE Transactions on Cognitive Communications and Networking*, vol. 11, no. 6, pp. 4240–4253, 2025.

[27] B. Zhu, K. Chi, J. Liu, K. Yu, and S. Mumtaz, "Efficient offloading for minimizing task computation delay of noma-based multiaccess edge computing," *IEEE Transactions on Communications*, vol. 70, no. 5, pp. 3186–3203, 2022.

[28] X. Zhou, C. Liu, and J. Zhao, "Resource Allocation of Federated Learning for the Metaverse with Mobile Augmented Reality," *IEEE Transactions on Wireless Communications*, 2023.

[29] X. Shang, "Enabling data-intensive workflows in heterogeneous edge-cloud networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 50, no. 3, pp. 36–38, 2023.

[30] M. R. Hossen and M. A. Islam, "Mobile task offloading under unreliable edge performance," *ACM SIGMETRICS Performance Evaluation Review*, vol. 48, no. 4, pp. 29–32, 2021.

[31] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-Minimization Algorithms in Signal Processing, Communications, and Machine Learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 794–816, 2016.

[32] Y. Li, W. Yu, and J. Zhao, "PrivTuner with Homomorphic Encryption and LoRA: A P3EFT Scheme for Privacy-Preserving Parameter-Efficient Fine-Tuning of AI Foundation Models," *arXiv preprint arXiv:2410.00433*, 2024.

[33] L. Su and J. Xu, "Securing Distributed Gradient Descent in High Dimensional Statistical Learning," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 1, pp. 1–41, 2019.

[34] C. T. Kelley, *Solving nonlinear equations with Newton's method*. SIAM, 2003.

[35] F. A. Potra and S. J. Wright, "Interior-point methods," *Journal of computational and applied mathematics*, vol. 124, no. 1-2, pp. 281–302, 2000.

[36] F. Flores-Bazán and G. Mastroeni, "Characterizing FJ and KKT Conditions in Nonconvex Mathematical Programming with Applications," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 647–676, 2015.

[37] K. Shen and W. Yu, "Fractional Programming for Communication Systems—Part I: Power Control and Beamforming," *IEEE Transactions on Signal Processing*, vol. 66, no. 10, pp. 2616–2630, 2018.

[38] ——, "Fractional Programming for Communication Systems—Part II: Uplink Scheduling via Matching," *IEEE Transactions on Signal Processing*, vol. 66, no. 10, pp. 2631–2644, 2018.

[39] Y. Jong, "An efficient global optimization algorithm for nonlinear sum-of-ratios problem," *Optimization Online*, pp. 1–21, 2012.

[40] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy Efficient Resource Allocation in UAV-Enabled Mobile Edge Computing Networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 9, pp. 4576–4589, 2019.

[41] Q. Wu, Y. Zeng, and R. Zhang, "Joint Trajectory and Communication Design for Multi-UAV Enabled Wireless Networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 2109–2121, 2018.

[42] F. Pervez, L. Zhao, and C. Yang, "Joint User Association, Power Optimization and Trajectory Control in an Integrated Satellite-Aerial-Terrestrial Network," *IEEE Transactions on Wireless Communications*, vol. 21, no. 5, pp. 3279–3290, 2021.

[43] C. Ding, J.-B. Wang, H. Zhang, M. Lin, and G. Y. Li, "Joint Optimization of Transmission and Computation Resources for Satellite and High Altitude Platform Assisted Edge Computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 2, pp. 1362–1377, 2021.

[44] C. Qiu, Z. Wei, X. Yuan, Z. Feng, and P. Zhang, "Multiple UAV-Mounted Base Station Placement and User Association With Joint Fronthaul and Backhaul Optimization," *IEEE Transactions on Communications*, vol. 68, no. 9, pp. 5864–5877, 2020.

[45] Y. Sun, J. Xu, and S. Cui, "User Association and Resource Allocation for MEC-Enabled IoT Networks," *IEEE Transactions on Wireless Communications*, vol. 21, no. 10, pp. 8051–8062, 2022.

[46] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, March 2004.