Network Utility Maximization with Unknown Utility Functions: A Distributed, Data-Driven Bilevel Optimization Approach

Kaiyi Ji
Department of CSE, University at Buffalo
Buffalo, USA
kaiyiji@buffalo.edu

Lei Ying
Department of EECS, University of Michigan
Ann Arbor, USA
leiying@umich.edu

ABSTRACT

Fair resource allocation is one of the most important topics in communication networks. Existing solutions almost exclusively assume each user utility function is known and concave. This paper seeks to answer the following question: how to allocate resources when utility functions are unknown, even to the users? This answer has become increasingly important in the next-generation AI-aware communication networks where the user utilities are complex and their closed-forms are hard to obtain. In this paper, we provide a new solution using a distributed and data-driven bilevel optimization approach, where the lower level is a distributed network utility maximization (NUM) algorithm with concave surrogate utility functions, and the upper level is a data-driven learning algorithm to find the best surrogate utility functions that maximize the sum of true network utility. The proposed algorithm learns from data samples (utility values or gradient values) to autotune the surrogate utility functions to maximize the true network utility, so works for unknown utility functions. For the general network, we establish the nonasymptotic convergence rate of the proposed algorithm with nonconcave utility functions. The simulations validate our theoretical results and demonstrate the great effectiveness of the proposed method in a real-world network.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Networks** → **Network algorithms**.

KEYWORDS

Network utility maximization, distributed bilevel optimization

ACM Reference Format:
Kaiyi Ji and Lei Ying. 2023. Network Utility Maximization with Unknown Utility Functions: A Distributed, Data-Driven Bilevel Optimization Approach. In *Proceedings of The Twenty-fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '23).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3565287.3610260

MobiHoc '23, October 23–26, 2023, Washington, DC, USA

ACM ISBN 978-1-4503-9926-5/23/10…$15.00
https://doi.org/10.1145/3565287.3610260

1 INTRODUCTION

Network utility maximization (NUM) has been studied for decades since the seminal work [21] and has been the central analytical framework for the design of fair and distributed resource allocation over the communication networks (e.g., the Internet, 6G networks). Its applications span from network congestion control [21, 28, 37], power allocation and routing in wireless networks [9, 34], load scheduling in cloud computing [10, 18, 31], to video streaming over dynamic networks [7, 12, 13, 23, 42, 49], and etc. A comprehensive introduction of the method and its connections to control theory and convex optimization can be found in [40].

In the traditional NUM, each user is associated with a utility function that captures the level of satisfaction with allocated resources (often the assigned data rate), and distributed NUM solutions and their variations have been implemented as the congestion control algorithms on the Internet, such as TCP-Reno, and scheduling algorithms for cellular networks, such as Proportional Fair Scheduling. The solutions maximize the total network utility subject to resource constraints such as channel capacity, average power, etc. There have been a large body of studies on NUM for wired [1, 3, 21, 25, 28, 37, 40, 47] and wireless networks [4, 6, 7, 12, 22, 23, 30, 33, 41, 45, 46, 49]. These existing studies on NUM almost exclusively assume that the utility functions are known to the users and are concave, e.g. the widely used α-fair utility functions [32], However, in many real-world applications, e.g., emerging AI-aware next-generation networks like 6G, the underlying utilities often correlate with user experience, information freshness, diversity, fidelity, job quality, etc, which can be nonconcave and generally unknown. Then, an open and challenging question in the field is:

How to allocate network resources fairly and efficiently when the utility functions are unknown and nonconcave?

The answer to this question has not been explored well except a few recent attempts [7, 44] using online learning algorithms. For example, [7] focused on a stochastic dynamic scenario, and proposed an online policy to gradually learn the utility functions and allocate resources accordingly. However, it still assumes the unknown utility functions are concave and requires a central scheduler.

In this paper, we consider unknown utility functions and provide a distributed solution from a new bilevel optimization perspective, where the lower-level problem is a standard distributed resource allocation algorithm with parameterized surrogate utility functions such as $\alpha-$fair utility functions, and the upper-level is to fine-tune the surrogate utility functions based on user experiences/feedback. While the solution is based on bilevel optimization, it is very different from existing studies for non-distributed bilevel optimization [5, 11, 14, 16, 17, 19, 27, 38] (see [26] and [15] for a more comprehensive overview) due to the distributed nature of

the solution over the communication networks. In addition, these approaches cannot be directly applied here due to the computation of either the Hessian inverse or a product of Hessians of the NUM objective, which requires each node to know the infeasible global network information, which is not practical. Although several decentralized bilevel optimization methods have been proposed by [2, 8, 29, 39, 48], they consider general bilevel objective functions without taking the channel capacity, the transmission links and the structured NUM objectives into account, and hence cannot be directly applied to the NUM problems. Then, the main contributions of this paper are summarized below:

- Our first contribution is the design of a distributed bilevel optimization algorithm named DBiNUM, which approximates the Hessian-inverse-vector product of the upper-level gradient using one-step gradient decent. We show that each user under DBiNUM only needs to know the partial network information such as transmission rates and link states of other users on her route, and hence DBiNUM admits a distributed implementation. In addition, DBiNUM does not need to know the true utility functions and only requires user feedback via gradient- or value-based queries.
- Theoretically, we prove that the hypergradient estimation error, although large initially, is formed by iteratively decreasing terms with a proper selection of the learning rates. Based on such key derivations, we provide the finite-time convergence rate guarantee for DBiNUM with a general nonconcave upper objective as well as a general network topology. We further provide a case study for a single-link multi-user network, where we show that when the true user utilities are α-fairness functions (but still unknown to the users), DBiNUM converges to the solution as if the utility functions are known. This provides some validation for the proposed bilevel formulation.
- In the simulations, we first validate our theoretical result by showing that our bilevel algorithm converges to the standard NUM solutions (total utility, user resources) when the true utility functions are α-fair utility functions. In a real-world Abilene network, we demonstrate that our bilevel approach achieves a significantly better network utility than the standard NUM baseline with fixed surrogate utility functions.

2 PROBLEM FORMULATION

Consider a communication network with n users (or data flows) and m communication links. Each user is associated with a utility function $\widetilde{U}_r(x_r)$, where x_r the transmission rate of user r. Let $\mathcal{L} = \{l_1, l_2, ..., l_m\}$ denote all communication links, c_l denote the capacity of link l, $\mathcal{L}_r$ denote all links along the route of user r, and $\mathbf{x} = [x_1, ..., x_n]^T$ denote the transmission rate vector. The network utility maximization (NUM) problem is to find a resource allocation $\mathbf{x}$ that solves the following optimization problem:

$$\begin{aligned} \max_{\mathbf{x}} \ & \sum_{r=1}^{n} \widetilde{U}_r(x_r) \\ \text{subject to:} \ & \sum_{r:l\in\mathcal{L}_r} x_r \le c_l, \quad \text{for any } l \in \mathcal{L} \\ & x_r \ge 0, \quad \text{for } r = 1, ..., n. \end{aligned} \tag{1}$$

Different from existing works on NUM, we consider general utility function $\widetilde{U}_r(x_r)$, not necessarily concave, and assume it may be unknown to user r.

2.1 The Traditional Network Utility Maximization and the Primal Solution

If $U_r(\cdot)$ is continuously twice differentiable and **concave**, e.g, α-fairness utility function such that $U_r(x_r) = \frac{x_r^{1-\alpha}}{1-\alpha}$, and is known to user r, then the problem in eq. (1) becomes the traditional NUM problem and has been extensively studied since the seminal work [21]. In particular, a variety of distributed algorithms have been proposed to solve eq. (1) efficiently with only limited information exchange between the user and the network. Among them, the primal approach penalizes the capacity constraints into the total network utility, and solves the following alternative regularized problem.

$$\min_{x_1,...,x_n>0} \quad \sum_{r=1}^{n} U_r(x_r) - \sum_{l\in\mathcal{L}} B_l\Big(\sum_{r:l\in\mathcal{L}_r} x_r\Big), \tag{2}$$

where the regularizer $B_l(\cdot)$ is continuously twice differentiable and μ-strongly-convex, and can be regarded as the cost of transmitting the data on link l to penalize the arrival rate for exceeding the link capacity. TCP-Reno for the Internet congestion control is such a primal algorithm.

2.2 NUM via Bilevel Optimization

The question we want to answer is how to solve NUM with unknown utility functions and how to solve it in a distributed fashion.We propose a distributed, bilevel solution to this problem. The lower level corresponds to a standard network resource allocation problem via a primal distributed algorithm as in eq. (2) with parameterized surrogate utility functions $U_r(x_r;\alpha_r)$, where $\boldsymbol{\alpha} \in \mathcal{A}$ are the parameters and the surrogate function is continuously twice differentiable and concave for any given $\boldsymbol{\alpha} \in \mathcal{A}$. The upper-level add-on procedure is to fine-tune the user-specified parameters $\alpha_r, r = 1, ..., n$ to learn the best surrogate utilities $U_r(x_r;\alpha_r), r = 1, ..., n$ based on the user feedback, e.g., the value-based query $\widetilde{U}_r(x_r)$ (i.e., how much the user feel satisfied with x_r) or the gradient-based query $\nabla\widetilde{U}_r(x_r)$ (i.e., how fast the user experience increases at x_r). Mathematically, this problem can be formulated as

$$\begin{aligned} &\max_{\boldsymbol{\alpha}\in\mathcal{A}} \left[\Psi(\boldsymbol{\alpha}) = \sum_{r=1}^{n} \widetilde{U}_r(x_r^*(\boldsymbol{\alpha}))\right] \\ &\mathbf{x}^*(\boldsymbol{\alpha}) = \arg\max_{\boldsymbol{x}>0} \Phi(\mathbf{x};\boldsymbol{\alpha}) = \sum_{r=1}^{n} \Big(U_r(x_r;\alpha_r) - \frac{\epsilon x_r^2}{2}\Big) \\ &\qquad - \sum_{l\in\mathcal{L}} B_l\Big(\sum_{i:l\in\mathcal{L}_i} x_i\Big), \end{aligned} \tag{3}$$

where $\mathcal{A} := \{\boldsymbol{\alpha} : \alpha_r \in \mathcal{A}_r, r = 1, ..., n\}$ is a closed, convex and bounded constraint set. Compared with eq. (2), we add a small quadratic term $-\frac{\epsilon x_r^2}{2}$ to each surrogate utility function $U_r(x_r;\alpha_r)$ to ensure that the lower-level objective function $\Phi(\boldsymbol{x};\boldsymbol{\alpha})$ is **strongly-concave** w.r.t. $\mathbf{x}$. Also note that this extra quadratic term changes the original solution of eq. (2) up to only an ϵ level, and hence the solution $\boldsymbol{x}^*(\boldsymbol{\alpha})$ is still valid.

3 ALGORITHM AND MAIN RESULTS

We first discuss the challenges in solving the bilevel problem eq. (3) and then present a distributed bilevel algorithm. We then provide the main results for the proposed method.

3.1 Challenges in Hypergradient Computation over Networks

Gradient ascent is a typical method to efficiently solve the bilevel problem in eq. (3). This process needs to calculate the gradient $\nabla\Psi(\boldsymbol{\alpha})$ (which we refer to the hypergradient) of the upper-level objective function. However, as shown in the following proposition, this hypergradient contains complicated components due to the nested problem structure.

Proposition 1. *Hypergradient $\nabla\Psi(\boldsymbol{\alpha})$ takes the form of*

$$\nabla\Psi(\boldsymbol{\alpha}) = -\nabla_{\boldsymbol{\alpha}}\nabla_{\boldsymbol{x}}\Phi(\boldsymbol{x}^*;\boldsymbol{\alpha})\big(\nabla^2_{\boldsymbol{x}}\Phi(\boldsymbol{x}^*;\boldsymbol{\alpha})\big)^{-1} \times \big[\nabla\widetilde{U}_1(x_1^*),...,\nabla\widetilde{U}_n(x_n^*)\big]^T, \tag{4}$$

where $\nabla_{\boldsymbol{\alpha}}\nabla_{\boldsymbol{x}}\Phi(\boldsymbol{x}^*;\boldsymbol{\alpha})$ is a diagonal matrix whose i^{th} diagonal element is $\nabla_\alpha\nabla_x U_i(x_i^*;\alpha_i)$, and the $(i,j)^{th}$ element of the Hessian matrix $\nabla^2_{\boldsymbol{x}}\Phi(\boldsymbol{x}^*;\boldsymbol{\alpha})$ equals to

$$\begin{cases}\nabla^2_x U_i(x_i^*;\alpha_i) - \epsilon - \sum_{l\in\mathcal{L}_i}\nabla^2 B_l\big(\sum_{r:l\in\mathcal{L}_r}x_r\big), i=j\\ -\sum_{l\in\mathcal{L}_i\cap\mathcal{L}_j}\nabla^2 B_l\big(\sum_{r:l\in\mathcal{L}_r}x_r\big),\ i\neq j,\end{cases} \tag{5}$$

where we define $\sum_{l\in\emptyset}(\cdot)=0$ for simplicity.

Note that the Hessian matrix $\nabla^2_{\boldsymbol{x}}\Phi(\boldsymbol{x}^*;\boldsymbol{\alpha})$ is invertible because the lower-level function $\Phi(\boldsymbol{x};\boldsymbol{\alpha})$ is strongly-concave. As shown in Proposition 1, the hypergradient $\nabla\Psi(\boldsymbol{\alpha})$ involves the second-order derivatives $\nabla_{\boldsymbol{\alpha}}\nabla_{\boldsymbol{x}}\Phi(\boldsymbol{x}^*;\boldsymbol{\alpha})$ and $\nabla^2_{\boldsymbol{x}}\Phi(\boldsymbol{x}^*;\boldsymbol{\alpha})$ of the lower-level function $\Phi(\boldsymbol{x}^*;\boldsymbol{\alpha})$. In particular, exactly computing $\nabla\Psi(\boldsymbol{\alpha})$ needs to invert the Hessian matrix $\nabla^2_{\boldsymbol{x}}\Phi(\boldsymbol{x}^*;\boldsymbol{\alpha})$ whose form is taken as in Proposition 1. However, this inversion is hard to implement in a large communication network because it requires the global network information but each user in reality knows only partial information. In addition, this inversion is computationally infeasible because the matrix dimension can be super large when the network contains millions of users. We next introduce a fast approximation method to tackle these two issues, which 1) allows a distributed implementation in the network and 2) is highly efficient without any Hessian inverse computations.

3.2 Proposed Distributed Bilevel Algorithm

In this section, we present a distributed bilevel method for solving the resource allocation problem in eq. (3).

As shown in algorithm 1, this algorithm involves a two-level optimization procedures. For the lower level, a standard distributed primal algorithm (examples can be found in [40]) is used to get δ_Φ-approximated solutions $\widehat{x}_{k,r}$ such that $|\widehat{x}_{k,r}-x^*_{k,r}|\le\delta_\Phi$ (δ_Φ is sufficiently small) under $\alpha_{k,r}$ for $r=1,...,n$, where $x^*_{k,r}, r=1,...,n$ are the lower-level solutions of the problem eq. (3) and are given by

$$x^*_{k,1},....,x^*_{k,n} = \underset{x_1,...,x_n>0}{\arg\max}\ \sum_{r=1}^{n}\big(U_r(x_r;\alpha_{k,r}) - \frac{\epsilon x_r^2}{2}\big) - \sum_{l\in\mathcal{L}}B_l\Big(\sum_{i:l\in\mathcal{L}_i}x_i\Big).$$

Note that the above solutions $\widehat{x}_{k,r}, r=1,...,n$ are achievable even in the presence of network delays as long as the execution time T_l is long enough [50].

For the next stage, all users continue to transmit packages to broadcast their information $\widehat{x}_{k,r}$ and $v_{k,r}$ for $r=1,...,n$ over the network. Each user stops broadcast once he receives all information $\widehat{x}_{k,i}$ from his neighbors $\mathcal{N}_r=\{i:\mathcal{L}_i\cap\mathcal{L}_r\neq\emptyset\}$ (including himself) and constraint-induced quantities $\nabla^2 B_l\big(\sum_{u:l\in\mathcal{L}_u}\widehat{x}_{k,u}\big)$ from all links $l\in\mathcal{L}_r$ along his path. This process is finished after a sufficiently long time T_o, i.e., no packages are transmitted in the networks. Note that each user can easily distinguish packages in this stage from those in the previous NUM procedure via identifying the existence of the new variable $v_{k,r}$.

After receiving the neighbor information $\widehat{x}_{k,i}, v_{k,i}$ for $i\in\mathcal{N}_r$, each user r update the auxiliary variable $v_{k,r}$ locally by

$$v_{k+1,r} = -\eta\sum_{i\in\mathcal{N}_r}\sum_{l\in\mathcal{L}_i\cap\mathcal{L}_r}\nabla^2 B_l\Big(\sum_{j:l\in\mathcal{L}_j}\widehat{x}_{k,j}\Big)v_{k,i} + \big(1-\epsilon+\eta\nabla^2_x U_r(\widehat{x}_{k,r};\alpha_{k,r})\big)v_{k,r} - \underbrace{\eta\nabla\widetilde{U}_r(\widehat{x}_{k,r})}_{\text{user feedback}}, \tag{6}$$

where the important quantity $\nabla\widetilde{U}_r(\widehat{x}_{k,r})$ reflects how fast the user experience can increase when increasing the current supply $\widehat{x}_{k,r}$. Note that the update in eq. (6) for user r only uses the information of its neighbors with at least one common link, so it is amenable to the practical decentralized implementation. The updates in eq. (6) for $r=1,...,n$ can be regarded as one-step approximation of the Hessian-inverse-vector $\big(\nabla^2_{\boldsymbol{x}}\Phi(\boldsymbol{x}^*_k;\boldsymbol{\alpha}_k)\big)^{-1}[\nabla\widetilde{U}_1(x^*_{k,1}),...,\nabla\widetilde{U}_n(x^*_{k,n})]^T$ of the hypergradient in eq. (4). The quantity $\nabla\widetilde{U}_r(\widehat{x}_{k,r})$ of eq. (6) is constructed via querying the use experience on the received

Algorithm 1 Distributed Bilevel Network Utility Maximization (DBiNUM)

1: **Input:** Initialization $\boldsymbol{a}_0\in\mathcal{A}$ and $\mathbf{x}_0>\mathbf{0}$
2: **for** $k=0,1,...,K$ **do**
3: Lower-level **standard** network maximization with T_l time slots:
- Use **standard** distributed primal algorithm to get $\widehat{x}_{k,r}\ge 0$ satisfying $|\widehat{x}_{k,r}-x^*_{k,r}|\le\delta_\Phi$ for each user r.

4: Information broadcast for upper level with T_o time slots:
- All users release packets with information $\widehat{x}_{k,r}$ and $v_{k,r}$ for $r=1,...,n$ for broadcast.
- Each user r collects $\widehat{x}_{k,i}$ from his neighbors $\mathcal{N}_r=\{i:\mathcal{L}_i\cap\mathcal{L}_r\neq\emptyset\}$ and $\nabla^2 B_l\big(\sum_{u:l\in\mathcal{L}_u}\widehat{x}_{k,u}\big)$ from its links $l\in\mathcal{L}_r$.

5: For each user r, update auxiliary variable $v_{k,r}$ by eq. (6).
6: For each user $r=1,...,n$, update user-specified parameters $\alpha_{k+1,r}$ by eq. (7).
7: **end for**

resource. As mentioned before, we use the gradient-type information from users to improve the resource allocation via asking how fast their experiences increase when supplying slightly more resource than $\widehat{x}_{k,r}$. In some circumstances where only utility values are observed, e.g., user satisfaction or job quality, we also provide a derivative-free gradient approximation using only utility values $\widetilde{U}_r(\cdot)$ in Section 6.

Finally, each user r updates the user-specified parameter $\alpha_{k,r}$ via a projected gradient ascend step as

$$\alpha_{k+1,r} = \mathcal{P}_{\mathcal{A}_r}\Big\{\alpha_{k,r} - \beta\nabla_\alpha\nabla_x U_r(\widehat{x}_{k,r};\alpha_{k,r})v_{k+1,r}\Big\}, \tag{7}$$

where $\beta > 0$ is the outer-loop stepsize and $\mathcal{P}_{\mathcal{A}_r}(\cdot)$ is the projection onto the constraint set $\mathcal{A}_r$.

3.3 Main Results

We present the finite-time convergence analysis for our proposed distributed method in Algorithm 1. All detailed proofs can be found in the arXiv version [20] of this submission. We first introduce some definitions and assumptions.

Definition 1. *$f(z) : \mathcal{Z} \to \mathbb{R}^d$ is L-Lipschitz continuous if for $\forall z_1, z_2 \in \mathcal{Z}$, $\|f(z_1) - f(z_2)\| \le L\|z_1 - z_2\|$.*

Without loss of generality, We make the following assumptions on the objective function in eq. (3).

Assumption 1. *The lower-level solution $\boldsymbol{x}^*(\boldsymbol{\alpha})$ in eq. (3) is bounded in the sense that there exist constants $\delta, b > 0$ such that its each coordinate satisfies $\delta < x_r^*(\boldsymbol{\alpha}) < b$, $r = 1, ..., n$ for $\forall\, \boldsymbol{\alpha} \in \mathcal{A}$.*

Assumption 1 says that the lower-level solutions $x_r^*, r = 1, ..., n$ are lower and upper-bounded by a small constant $\delta > 0$ and a sufficiently large constant b. This assumption is reasonable because the regularization $B_l(\cdot)$ prevents the solutions from converging to the infinity and the lower bound constant δ helps to avoid some trouble when $x_r \to 0$ for some utility function such as $\log(x_r)$ and $\frac{x_r^{1-\alpha_r}}{1-\alpha_r}$ with $\alpha_r > 1$. For example, it can be shown that the solutions of for α-fairness utility function $U_r(x_r;\alpha_r) = \frac{x_r^{1-\alpha_r}}{1-\alpha_r}$ satisfies Assumption 1 given the boundedness of $\boldsymbol{\alpha} \in \mathcal{A}$.

The following assumption imposes some geometrical conditions on the utility function $U_r(\cdot\,;\alpha_r)$ and the regularization function $B_l(\cdot)$. Let $\mathcal{X} := \{\boldsymbol{x} : \frac{\delta}{2} < x_r < 2b, r = 1, ..., n\}$.

Assumption 2. *For any $\boldsymbol{\alpha} \in \mathcal{A}$ and any $\boldsymbol{x} \in \mathcal{X}$,*

- *$U_r(\cdot\,;\alpha_r)$ is concave and $B_l(\cdot)$ is μ-strongly-convex.*
- *$\widetilde{U}_r(\cdot)$, $\nabla\widetilde{U}_r(\cdot)$, $\nabla_x U_r(\cdot\,;\cdot)$, $\nabla_\alpha\nabla_x U_r(\cdot\,;\cdot)$, $\nabla_x^2 U_r(\cdot\,;\cdot)$ are L_u-Lipschitz continuous.*
- *$\nabla B_l(\cdot)$ and $\nabla^2 B_l(\cdot)$ are L_b-Lipschitz continuous.*

Assumption 2 cover many utility functions of practical interest such as log utility $\log(x_r)$ and α-fairness utility, as well as a variety of regularizers such as the quadratic function $\frac{\mu}{2}x^2$ and the barrier function $-\log(c-x)$ for $\frac{\delta}{2} < x < 2b < c$. For example, for the α-fairness utility function $\frac{x_r^{1-\alpha_r}}{1-\alpha_r}$, the Lipschitz continuity assumption holds because its high-order derivatives such as $\nabla_x U_r(x_r;\alpha_r) = \frac{1}{x_r^{\alpha_r}}$, $\nabla_x^2 U_r(x_r;\alpha_r) = \frac{-\alpha_r}{x_r^{\alpha_r+1}}$, $\nabla_x^3 U_r(x_r;\alpha_r) = \frac{\alpha_r(\alpha_r+1)}{x_r^{\alpha_r+2}}$ are **bounded** due to the boundedness of $\alpha_r \in \mathcal{A}_r$ and $x_r \in (\frac{\delta}{2}, 2b)$. The following theorem characterizes the convergence rate analysis for the proposed algorithm with general utility functions and networks.

Theorem 1. *Suppose Assumptions 1 and 2 hold. Choose $\delta_\phi < \frac{\delta}{2}$, $\eta < \frac{1}{L_{grad}}$ and $\beta \le \min\big(\sqrt{\frac{\eta\mu_\Phi}{256C_v L_u^2}}, \frac{1}{2L_\Psi}\big)$, where $L_\Psi = \big(\frac{L_{grad}}{\mu_\Phi}\big(\frac{\sqrt{n}L_u^2}{\mu_\Phi} + \frac{\sqrt{n}L_u^2 L_{Hess}}{\mu_\Phi^2} + \frac{L_u}{\mu_\Phi}\big) + \frac{\sqrt{n}L_u^2}{\mu_\Phi} + \frac{\sqrt{n}L_u^3}{\mu_\Phi^2}\big)$ is the smoothness constant of the total objective function $\Psi(\boldsymbol{\alpha})$. Then, the iterates generated by Algorithm 1 satisfy*

$$\frac{1}{K}\sum_{k=0}^{K-1}\|G_{proj}(\boldsymbol{\alpha}_k)\|^2 \le \underbrace{\frac{16(\max_{\boldsymbol{\alpha}\in\mathcal{A}}\Psi(\boldsymbol{\alpha}) - \Psi(\boldsymbol{\alpha}_0))}{\beta K} + \frac{256nL_u^4(1+\mu_\Phi^2)}{\eta\mu_\Phi^3}\frac{1}{K}}_{\text{Sublinearly decaying terms}} + \underbrace{\frac{128L_u^2C_\Phi\delta_\Phi^2}{\eta\mu_\Phi} + \frac{4L_u^4n^2\delta_\Phi^2}{\mu_\Phi^2}}_{\text{Lower-level error}},$$

where $G_{proj}(\boldsymbol{\alpha}_k) = \beta^{-1}(\mathcal{P}_{\mathcal{A}}\{\boldsymbol{\alpha}_k + \beta\nabla\Psi(\boldsymbol{\alpha}_k)\} - \boldsymbol{\alpha}_k)$ denote the generalized projected gradient at the k^{th} iteration, and μ_Φ, L_{grad}, L_{Hess}, constants C_Φ and C_v are defined in Propositions 2, 3 and 4.

Theorem 1 uses the generalized gradient $G_{\text{proj}}(\boldsymbol{\alpha}_k)$ instead of the gradient $\nabla\Psi(\boldsymbol{\alpha}_k)$ due to the existence of the projection. Note that if the iterate $\boldsymbol{\alpha}_k + \beta\nabla\Psi(\boldsymbol{\alpha}_k)$ locates inside of the constraint set $\mathcal{A}$, this generalized gradient $G_{\text{proj}}(\boldsymbol{\alpha}_k)$ reduces to the vanilla gradient $\nabla\Psi(\boldsymbol{\alpha}_k)$. Theorem 1 shows that the proposed DBiNUM finds a stationary point $\boldsymbol{\alpha}_s$ with $s = \arg\min_k \|G_{\text{proj}}(\boldsymbol{\alpha}_k)\|^2$ for the constrained nonconcave bilevel problem in eq. (3), whose generalized projected gradient norm $\|G_{\text{proj}}(\boldsymbol{\alpha}_s)\|^2$ contains a sublinearly decaying term and a convergence error $\frac{128L_u^2C_\Phi\delta_\Phi^2}{\eta\mu_\Phi} + \frac{4L_u^4n^2\delta_\Phi^2}{\mu_\Phi^2}$ induced by the approximation error δ_Φ of the lower-level network utility maximization. This convergence error can be arbitrarily small by setting the lower-level target accuracy δ_Φ small, e.g., at an ϵ accuracy. Note that we adopt the stationary point as the convergence criterion due to the general nonconcavity of the upper-level objective function $\widetilde{U}_r$.

4 PROOF OF THE MAIN RESULT

In this section, we provide the technical proofs for Theorem 1. We first prove an important strongly-concave geometry of the lower-level objective function $\Phi(\mathbf{x};\boldsymbol{\alpha})$.

Proposition 2. *Suppose Assumptions 2 holds. For any $\boldsymbol{\alpha} \in \mathcal{A}, \boldsymbol{x} \in \mathcal{X}$, $\Phi(\boldsymbol{x}\,;\boldsymbol{\alpha})$ is μ_Φ-strongly-concave w.r.t. $\boldsymbol{x}$, where $\mu_\Phi := \frac{\epsilon+\mu M_{\min}}{2}$ and the topology-related constant $M_{\min}$ is defined as*

$$M_{\min} = \min_{r=1,...,n}\{M_r : \textit{number of links user } r \textit{ exclusively occupies}\}.$$

Note that the strong-concavity constant $\frac{\epsilon+\mu M_{\min}}{2}$ depends on the network topology due to the factor $M_{\min}$. For the case where each user r occupies solely at least one link, $M_{\min} \ge 1$ and hence the quadratic term $\frac{\epsilon}{2}x_r^2, r = 1, ..., n$ in eq. (3) are not needed. However,

for the general topology, this quadratic regularization is necessary to guarantee the strong-concavity.

In the worst cases, the smoothness parameter of the Hessian matrix $\nabla^2_{\boldsymbol{x}}\Phi(\cdot;\boldsymbol{\alpha})$ whose form is given by Proposition 1 scales in the order of $n^{\frac{3}{2}}|\mathcal{L}|$, which can be prohibitively large in the network with millions of users and links, and hence leads to slow convergence in practice. For this reason, we next provide a refined analysis of the smoothness of quantities $\nabla^2_{\boldsymbol{x}}\Phi(\boldsymbol{x};\boldsymbol{\alpha})$ and $\nabla_{\boldsymbol{\alpha}}\nabla_{\boldsymbol{x}}\Phi(\boldsymbol{x};\boldsymbol{\alpha})$ by taking the sparse network structure (i.e., each user shares links with only some of other users) into account.

Proposition 3. *Suppose Assumption 2 holds. Then, for any $\boldsymbol{\alpha}\in\mathcal{A}$, $\boldsymbol{x}\in\mathcal{X}$ and any vector $\boldsymbol{u}=[u_1,...,u_n]$,*

$$\begin{aligned}
\|\nabla_{\boldsymbol{x}}\Phi(\boldsymbol{x};\boldsymbol{\alpha})-\nabla_{\boldsymbol{x}}\Phi(\boldsymbol{x}';\boldsymbol{\alpha})\| &\le L_{grad}\|\boldsymbol{x}-\boldsymbol{x}'\|,\\
\|\nabla^2_{\boldsymbol{x}}\Phi(\boldsymbol{x};\boldsymbol{\alpha})\boldsymbol{u}-\nabla^2_{\boldsymbol{x}}\Phi(\boldsymbol{x}';\boldsymbol{\alpha})\boldsymbol{u}\| &\le L_{Hess}\max_i|u_i|\|\boldsymbol{x}-\boldsymbol{x}'\|,\\
\|\nabla^2_{\boldsymbol{x}}\Phi(\boldsymbol{x};\boldsymbol{\alpha})\boldsymbol{u}-\nabla^2_{\boldsymbol{x}}\Phi(\boldsymbol{x};\boldsymbol{\alpha}')\boldsymbol{u}\| &\le L_u\max_i|u_i|\|\boldsymbol{\alpha}-\boldsymbol{\alpha}'\|,
\end{aligned}$$

where the constants

$$L_{grad}=\sqrt{2L_u^2+2n\sum_{i=1}^{n}\sum_{l:l\in\mathcal{L}_i}L_b^2}$$

$$L_{Hess}=\sqrt{2L_u^2+2nL_b^2\max_i\Big(\sum_{j:\mathcal{L}_i\cap\mathcal{L}_j\neq\emptyset}\sum_{l\in\mathcal{L}_i\cap\mathcal{L}_j}1\Big)^2}$$

are related to the network topology. Similarly, for the mixed derivative $\nabla_{\boldsymbol{\alpha}}\nabla_{\boldsymbol{x}}\Phi(\boldsymbol{x};\boldsymbol{\alpha})$, we have

$$\begin{aligned}
\|\nabla_{\boldsymbol{\alpha}}\nabla_{\boldsymbol{x}}\Phi(\boldsymbol{x};\boldsymbol{\alpha})\boldsymbol{u}-\nabla_{\boldsymbol{\alpha}}\nabla_{\boldsymbol{x}}\Phi(\boldsymbol{x}';\boldsymbol{\alpha})\boldsymbol{u}\| &\le L_u\max_i|u_i^2|\|\boldsymbol{x}-\boldsymbol{x}'\|,\\
\|\nabla_{\boldsymbol{\alpha}}\nabla_{\boldsymbol{x}}\Phi(\boldsymbol{x};\boldsymbol{\alpha})\boldsymbol{u}-\nabla_{\boldsymbol{\alpha}}\nabla_{\boldsymbol{x}}\Phi(\boldsymbol{x};\boldsymbol{\alpha}')\boldsymbol{u}\| &\le L_u\max_i|u_i|\|\boldsymbol{\alpha}-\boldsymbol{\alpha}'\|.
\end{aligned}$$

It can be observed from Proposition 3 that the smoothness constant of $\nabla^2_{\boldsymbol{x}}\Phi(\cdot;\boldsymbol{\alpha})\boldsymbol{u}$ scales in the order of

$$\sqrt{n\max_i(\sum_{j:\mathcal{L}_i\cap\mathcal{L}_j\neq\emptyset}\sum_{l\in\mathcal{L}_i\cap\mathcal{L}_j}1)^2},$$

which represents to the total number of links the users i share with other users. As mentioned before, In the worst case, i.e., all users share the same links, this constant takes the order of $n^{\frac{3}{2}}|\mathcal{L}|$. However, in the practical network, each user shares links with a small portion of users, and hence each $\sum_{j:\mathcal{L}_i\cap\mathcal{L}_j\neq\emptyset}\sum_{l\in\mathcal{L}_i\cap\mathcal{L}_j}1$ is much smaller than the worst-case $n|\mathcal{L}|$. We next characterize the error in approximating the Hessian-inverse-vector product in the hypergradient at iteration k. For notational convenience, let $\boldsymbol{v}_k=[v_{k,1},...,v_{k,n}]^T$ and $\nabla\widetilde{U}(\boldsymbol{x})=\big[\nabla\widetilde{U}_1(x_1),...,\nabla\widetilde{U}_n(x_n)\big]^T$.

Proposition 4. *Suppose Assumptions 1 and 2 hold. Choose $\delta_\phi<\frac{\delta}{2}$ and $\eta<\frac{1}{L_{grad}}$. Let $C_v=n\big(1+\frac{2}{\eta\mu_\Phi}\big)\big(\frac{L_{grad}}{\mu_\Phi}\big(\frac{L_uL_{Hess}}{\mu_\Phi^2}+\frac{L_u}{\sqrt{n}\mu_\Phi}\big)+\frac{L_u^2}{\mu_\Phi^2}\big)^2$ and $C_\Phi=4\big(1+\frac{1}{\eta\mu_\Phi}\big)\big(\frac{L_{Hess}L_u}{\mu_\Phi^2}+\frac{L_u}{\sqrt{n}\mu_\Phi}\big)^2n^2$. Then, we have*

$$\begin{aligned}
&\|\boldsymbol{v}_{k+1}-(\nabla^2_{\boldsymbol{x}}\Phi(\boldsymbol{x}_k^*;\boldsymbol{\alpha}_k))^{-1}\nabla\widetilde{U}(\boldsymbol{x}_k^*)\|^2\\
&\le\big(1-\frac{\eta\mu_\Phi}{2}\big)\big\|\boldsymbol{v}_k-\nabla^2_{\boldsymbol{x}}\Phi(\boldsymbol{x}_{k-1}^*;\boldsymbol{\alpha}_{k-1})^{-1}\nabla\widetilde{U}(\boldsymbol{x}_{k-1}^*)\big\|^2\\
&+C_v\|\boldsymbol{\alpha}_k-\boldsymbol{\alpha}_{k-1}\|^2+C_\Phi\delta_\Phi^2. \qquad (8)
\end{aligned}$$

Proposition 4 characterizes the error of $\boldsymbol{v}_{k+1}$ in approximating the Hessian-inverse-vector product $(\nabla^2_{\boldsymbol{x}}\Phi(\boldsymbol{x}_k^*;\boldsymbol{\alpha}_k))^{-1}\nabla\widetilde{U}(\boldsymbol{x}_k^*)$ of the hypergradient. It can be seen from eq. (8) that this error contains an iteratively decreasing term (i.e., the first term at the right hand side) and two error terms $C_v\|\boldsymbol{\alpha}_k-\boldsymbol{\alpha}_{k-1}\|$ (which captures the difference between two adjacent iterations) and $C_\Phi\delta_\Phi^2$ (which is induced by the lower-level estimation error $\|\widehat{\boldsymbol{x}}_k-\boldsymbol{x}_k\|$). By choosing the upper-level stepsize β small enough, we can well control the increment $\|\boldsymbol{\alpha}_k-\boldsymbol{\alpha}_{k-1}\|$ and guarantee the hypergradient estimation error not to explode. Based on the form of the hypergradient established in Proposition 1, the update in eq. (7) can be written as

$$\boldsymbol{\alpha}_{k+1}=\mathcal{P}_{\mathcal{A}}\big\{\boldsymbol{\alpha}_k-\beta\nabla_{\boldsymbol{\alpha}}\nabla_{\boldsymbol{x}}\Phi(\widehat{\boldsymbol{x}}_k;\boldsymbol{\alpha}_k)\boldsymbol{v}_{k+1}\big\}, \qquad (9)$$

where $\widehat{\nabla}\Psi(\boldsymbol{\alpha}_k):=-\nabla_{\boldsymbol{\alpha}}\nabla_{\boldsymbol{x}}\Phi(\widehat{\boldsymbol{x}}_k;\boldsymbol{\alpha}_k)\boldsymbol{v}_{k+1}$ serves as an estimator of the hypergradient $\nabla\Psi(\boldsymbol{\alpha}_k)$ given by eq. (4). We now characterize the error between $\widehat{\nabla}\Psi(\boldsymbol{\alpha}_k)$ and $\nabla\Psi(\boldsymbol{\alpha}_k)$.

Proposition 5. *Suppose Assumptions 1 and 2 hold. Choose $\delta_\phi<\frac{\delta}{2},\eta<\frac{1}{L_{grad}}$ and $\beta<\sqrt{\frac{\eta\mu_\Phi}{16C_vL_u^2}}$. Then,*

$$\begin{aligned}
&\big\|\widehat{\nabla}\Psi(\boldsymbol{\alpha}_k)-\nabla\Psi(\boldsymbol{\alpha}_k)\big\|^2\\
&\le\big(1-\frac{\eta\mu_\Phi}{4}\big)^k4nL_u^4(1+\mu_\Phi^{-2})\\
&+4C_vL_u^2\beta^2\sum_{t=0}^{k-1}\big(1-\frac{\eta\mu_\Phi}{4}\big)^{k-1-t}\|G_{proj}(\boldsymbol{\alpha}_t)\|^2\\
&+\frac{8L_u^2C_\Phi\delta_\Phi^2\mu_\Phi+4\eta L_u^4n^2\delta_\Phi^2}{\eta\mu_\Phi^2}, \qquad (10)
\end{aligned}$$

where the constants C_v,C_Φ are given in Proposition 4,

Proposition 5 shows that the bound on the hypergradient estimation error $\big\|\widehat{\nabla}\Psi(\boldsymbol{\alpha}_k)-\nabla\Psi(\boldsymbol{\alpha}_k)\big\|^2$ contains three terms, i.e., an exponentially decaying term, an error term proportional to the average gradient norm, and a sufficiently small error term induced by the lower-level approximation. Based on the results in Propositions 2, 3,4 and 5, we now characterize the convergence rate performance of the distributed bilevel method in Algorithm 1.

Proof Sketch of Theorem 1. The first step is to derive the smoothness property of the hypergradient $\nabla\Psi(\cdot)$. Based on the form of $\nabla\Psi(\cdot)$ in eq. (4) and using Proposition 2, Proposition 3, we have, for any two $\boldsymbol{\alpha}_1,\boldsymbol{\alpha}_2\in\mathcal{A}$

$$\begin{aligned}
&\|\nabla\Psi(\boldsymbol{\alpha}_1)-\nabla\Psi(\boldsymbol{\alpha}_2)\|\\
&\le L_u(\|\boldsymbol{x}^*(\boldsymbol{\alpha}_1)-\boldsymbol{x}^*(\boldsymbol{\alpha}_2)\|+\|\boldsymbol{\alpha}_1-\boldsymbol{\alpha}_2\|)\frac{\sqrt{n}L_u}{\mu_\Phi}\\
&+\frac{\sqrt{n}L_u^2}{\mu_\Phi^2}\big(L_{\rm Hess}\|\boldsymbol{x}^*(\boldsymbol{\alpha}_1)-\boldsymbol{x}^*(\boldsymbol{\alpha}_2)\|+L_u\|\boldsymbol{\alpha}_1-\boldsymbol{\alpha}_2\|\big)\\
&+\frac{L_u}{\mu_\Phi}\|\boldsymbol{x}^*(\boldsymbol{\alpha}_1)-\boldsymbol{x}^*(\boldsymbol{\alpha}_2)\|, \qquad (11)
\end{aligned}$$

which, using $\|\boldsymbol{x}^*(\boldsymbol{\alpha}_1)-\boldsymbol{x}^*(\boldsymbol{\alpha}_2)\|\le\frac{L_{\rm grad}}{\mu_\Phi}\|\boldsymbol{\alpha}_1-\boldsymbol{\alpha}_2\|$, yields

$$\|\nabla\Psi(\boldsymbol{\alpha}_1)-\nabla\Psi(\boldsymbol{\alpha}_2)\|\le L_\Psi\|\boldsymbol{\alpha}_1-\boldsymbol{\alpha}_2\|. \qquad (12)$$

Let $\widehat{\nabla}\Psi(\boldsymbol{\alpha}_k)=-\nabla_{\boldsymbol{\alpha}}\nabla_{\boldsymbol{x}}\Phi(\widehat{\boldsymbol{x}}_k;\boldsymbol{\alpha}_k)\boldsymbol{v}_{k+1}$ demote the hypergradient estimate. Then, based on the smoothness property established in

eq. (12), we have

$$\begin{aligned}\Psi(\boldsymbol{\alpha}_{k+1}) - \Psi(\boldsymbol{\alpha}_k) &\geq \left\langle \widehat{\nabla}\Psi(\boldsymbol{\alpha}_k), \mathcal{P}_{\mathcal{A}}\left\{\boldsymbol{\alpha}_k + \beta\widehat{\nabla}\Psi(\boldsymbol{\alpha}_k)\right\} - \boldsymbol{\alpha}_k \right\rangle \\ &\quad + \left\langle \nabla\Psi(\boldsymbol{\alpha}_k) - \widehat{\nabla}\Psi(\boldsymbol{\alpha}_k), \mathcal{P}_{\mathcal{A}}\left\{\boldsymbol{\alpha}_k + \beta\widehat{\nabla}\Psi(\boldsymbol{\alpha}_k)\right\} - \boldsymbol{\alpha}_k \right\rangle \\ &\quad - \frac{L_\Psi}{2}\|\boldsymbol{\alpha}_{k+1} - \boldsymbol{\alpha}_k\|^2. \end{aligned} \tag{13}$$

Using the property of the projection on the convex set $\mathcal{A}$, i.e., $\langle \boldsymbol{x} - \mathcal{P}_{\mathcal{A}}(\boldsymbol{x}), \boldsymbol{y} - \mathcal{P}_{\mathcal{A}}(\boldsymbol{x})\rangle \leq 0$ for any $\boldsymbol{y} \in \mathcal{A}$ and noting that $\boldsymbol{\alpha}_k \in \mathcal{A}$, the first term of the right hand side of eq. (13) can be lower-bounded by

$$\frac{1}{\beta}\|\boldsymbol{\alpha}_k - \mathcal{P}_{\mathcal{A}}\left\{\boldsymbol{\alpha}_k + \beta\widehat{\nabla}\Psi(\boldsymbol{\alpha}_k)\right\}\|^2. \tag{14}$$

Let $\widehat{G}_{\text{proj}}(\boldsymbol{\alpha}_k) = \beta^{-1}\left(\mathcal{P}_{\mathcal{A}}\left\{\boldsymbol{\alpha}_k + \beta\widehat{\nabla}\Psi(\boldsymbol{\alpha}_k)\right\} - \boldsymbol{\alpha}_k\right)$ be the estimate of the generalized projected gradient $G_{\text{proj}}(\boldsymbol{\alpha}_k)$ defined in Proposition 5. Then, substituting eq. (14) into eq. (13) and based on $\langle \boldsymbol{a}, \boldsymbol{b}\rangle \geq -\frac{1}{2}(\|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2)$ and the non-expansive property of the projection on convex sets, we have

$$\begin{aligned}\Psi(\boldsymbol{\alpha}_{k+1}) \geq& \Psi(\boldsymbol{\alpha}_k) + \left(\frac{\beta}{4} - \frac{L_\Psi\beta^2}{4}\right)\|G_{\text{proj}}(\boldsymbol{\alpha}_k)\|^2 \\ &- \left(\beta - \frac{L_\Psi\beta^2}{2}\right)\|\nabla\Psi(\boldsymbol{\alpha}_k) - \widehat{\nabla}\Psi(\boldsymbol{\alpha}_k)\|^2. \end{aligned} \tag{15}$$

Applying Proposition 5 to the above eq. (15), conducting the telescoping and using the fact that $\sum_{k=1}^{K-1}\sum_{t=0}^{k-1} a_{k-1-t}b_t \leq \sum_{k=0}^{K-1} a_k \sum_{t=0}^{K-1} b_t$ for $a_t, b_t \geq 0$, we have

$$\begin{aligned}&\left(\frac{1}{8} - \frac{16C_o L_u^2\beta^2}{\eta\mu_\Phi}\right)\frac{1}{K}\sum_{k=0}^{K-1}\|G_{\text{proj}}(\boldsymbol{\alpha}_k)\|^2 \\ &\leq \frac{\max_{\boldsymbol{\alpha}\in\mathcal{A}}\Psi(\boldsymbol{\alpha}) - \Psi(\boldsymbol{\alpha}_0)}{\beta K} \\ &\quad + \frac{16nL_u^4(1+\mu_\Phi^{-2})}{\eta\mu_\Phi}\frac{1}{K} + \frac{8L_u^2C_\Phi\delta_\Phi^2\mu_\Phi + 4\eta L_u^4 n^2\delta_\Phi^2}{\eta\mu_\Phi^2}, \end{aligned}$$

which, in conjunction with the choice of $\beta \leq \sqrt{\frac{\eta\mu_\Phi}{256C_oL_u^2}}$, completes the proof. □

5 VALIDATION STUDY OF BILEVEL FORMULATION

In this section, we provide a case study for a single-link multi-user network as shown in Figure 1 to validate the bilevel formulation we propose in eq. (3), where all n users share the same communication link with a capacity P. In this setting, the bilevel formulation is solve the following problem.

$$\begin{aligned}&\max_{\boldsymbol{\alpha}\in\mathcal{A}} \quad \Psi(\boldsymbol{\alpha}) = \sum_{r=1}^n \frac{x_r^*(\boldsymbol{\alpha})^{1-\widetilde{\alpha}_r}}{1-\widetilde{\alpha}_r}, \\ &x_1^*(\boldsymbol{\alpha}), ..., x_n^*(\boldsymbol{\alpha}) = \underset{x_r>0, \sum_{r=1}^n x_r \leq P}{\arg\max} \sum_{r=1}^n \frac{x_r^{1-\alpha_r}}{1-\alpha_r}, \end{aligned} \tag{16}$$

where we adopt a simple bounded constraint set $\mathcal{A} := \{0 < a_r \leq b, r = 1, ..., n\}$. Note that the lower level adopts the original problem in eq. (1) rather than the primal version as in eq. (3) because the explicit solutions can be obtained here.

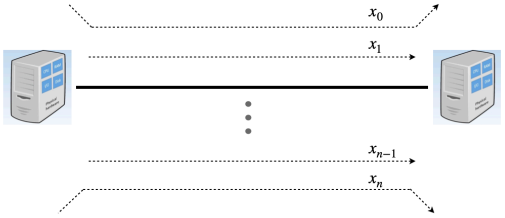

Figure 1: Example: single communication link with n users.

The following theorem establishes the equivalence between the solutions of the bilevel problem in eq. (16) and the standard single-level NUM, when the true user utilities are α-fairness functions.

Theorem 2. *Let $\boldsymbol{\alpha}^* \in \arg\max_{\boldsymbol{\alpha}\in\mathcal{A}}\Psi(\boldsymbol{\alpha})$ be any solution of the bilevel problem in eq. (16). Then, the resulting allocated resources $x_r^*(\boldsymbol{\alpha}^*), r = 1, ..., n$ from the bilevel formulation recover the solutions of the following standard utility maximization problem under α-fairness utilities with fixed parameters $\widetilde{\alpha}_r > 0, r = 1, ..., n$.*

$$\max_{x_1,...,x_n} \sum_{r=1}^n \left[\widetilde{U}_r(x_r;\widetilde{\alpha}_r) = \frac{x_r^{1-\widetilde{\alpha}_r}}{1-\widetilde{\alpha}_r}\right], \tag{17}$$

subject to $\sum_{r=1}^n x_r \leq P$ and $x_r > 0$, for $r = 1, ..., n$.

Theorem 2 shows that the solution $\boldsymbol{x}^*(\boldsymbol{\alpha}^*)$ of the bilevel problem we formulate in eq. (16) also maximizes the original network utility maximization problem in eq. (17). This means that the proposed DBiNUM converges to a solution as if the utility functions are known. This case study provides some validation of the proposed bilevel objective function. We note that our analysis is possibly extended to the multi-link scenarios with the graph structure satisfying certain properties.

6 DISCUSSION ON USER FEEDBACK

It can be seen from eq. (6) that DBiNUM takes the user (or application) information $\nabla\widetilde{U}_r(\widehat{x}_{k,r})$ to improve the selection of the user utility functions. In other words, each user has to give feedback to the network showing how fast their experiences increase at the given allocated resource $\widehat{x}_{k,r}$. However, in some circumstances, only utility values are available such as energy consumption, user satisfaction or job quality, and hence a more feasible solution is to query their utility value at x, i.e., $\widetilde{U}_r(x)$. Given such value information, one can use a gradient-free approach to approximate the gradient $\nabla\widetilde{U}_r(\widehat{x}_{k,r})$ by taking the utility difference at two close points $\widehat{x}_{k,r}$ and $\widehat{x}_{k,r} + \delta u$, as shown below.

$$\widehat{\nabla}_{\text{two}}\widetilde{U}_r(\widehat{x}_{k,r};u) = \frac{\widetilde{U}_r(\widehat{x}_{k,r}+\delta u) - \widetilde{U}_r(\widehat{x}_{k,r})}{\delta}u, \tag{18}$$

where $\delta > 0$ is the smoothing parameter and u is a standard Gaussian random variable. Based on the results in [36], it can be shown that the estimation bias $\left|\mathbb{E}_u\widehat{\nabla}_{\text{two}}\widetilde{U}_r(\widehat{x}_{k,r};u) - \nabla\widetilde{U}_r(\widehat{x}_{k,r})\right|$ of the above two-point estimator is bounded by $4L_u\delta$, which can be small by choosing a small δ. Hence, we can establish a convergence rate result similar to Theorem 1 with an error proportional to δ.

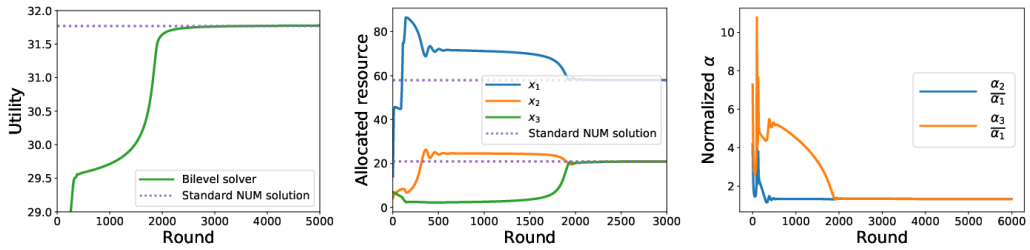

Figure 2: Network utility maximization via our proposed bilevel solver DBiNUM in a 3-user setting. Left plot: total underlying utility Ψ v.s. # of rounds; middle plot: allocated resource v.s. # of rounds; right plot: normalized α v.s. # of rounds.

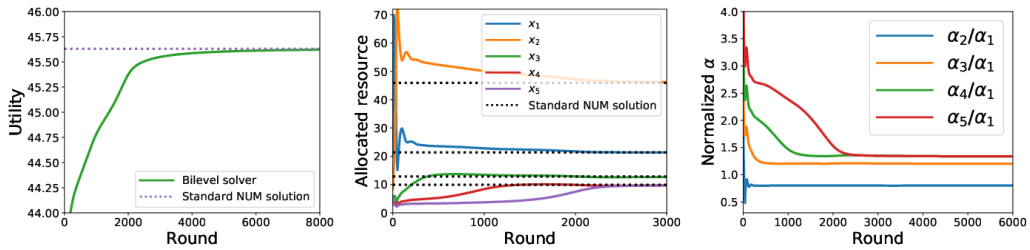

Figure 3: Network utility maximization via our proposed bilevel solver DBiNUM in a 5-user setting. Left plot: total underlying utility Ψ v.s. # of rounds; middle plot: allocated resource v.s. # of rounds; right plot: normalized α v.s. # of rounds.

Note that the estimator in eq. (18) requires to query the utility value $\widetilde{U}_r(\cdot)$ at two points simultaneously. However, in the time-varying and non-stationary environments, $\widetilde{U}_r$ is changing with time, and hence the two-point estimator may contain large estimation error. In this case, one-point approach turns out to be more appealing, which takes the form of

$$\widehat{\nabla}_{\text{one}}\widetilde{U}_r(\widehat{x}_{k,r};u)=\frac{\widetilde{U}_r(\widehat{x}_{k,r}+\delta u)u}{\delta}. \tag{19}$$

Note that the above one-query estimator has the same mean as the two-query estiamtion, i.e., $\mathbb{E}_u\widehat{\nabla}_{\text{one}}\widetilde{U}_r(\widehat{x}_{k,r};u)=\mathbb{E}_u\widehat{\nabla}_{\text{two}}\widetilde{U}_r(\widehat{x}_{k,r};u)$, so the convergence analysis in Theorem 1 is still applied.

7 DISCUSSION ON LOWER-LEVEL METHOD

Our method can be regarded as adding a top-level procedure over a lower-level standard network resource allocation process to improve the overall network utility. In this section, we discuss the impact of the lower-level procedure on our convergence analysis.

In Algorithm 1, the lower-level procedure adopts a distributed primal solution (see [40]) given by $\mathbf{x}^*(\boldsymbol{\alpha})=\arg\max_{\boldsymbol{x}}\Phi(\mathbf{x};\boldsymbol{\alpha})=\sum_{r=1}^n\left(U_r(x_r;\alpha_r)-\frac{\epsilon x_r^2}{2}\right)-\sum_{l\in\mathcal{L}}B_l\big(\sum_{i:l\in\mathcal{L}_i}x_i\big)$, as given in eq. (3). To solve this objective with given α_r, each user first computes the gradient information $\nabla_x U_r(x_r;\alpha_r)-\epsilon x_r-\sum_{l\in\mathcal{L}_r}\nabla B_l(\sum_{i:l\in\mathcal{L}_i}x_i)$ using the information from his neighbors with shared links, and then run simple gradient-based updates. It has been shown in Propositions 2 and 3 that the lower-level function $\Phi(\boldsymbol{x};\boldsymbol{\alpha})$ is strongly-convex and smooth w.r.t. $\boldsymbol{x}$, respectively. Then, based on the results for smooth convex optimization [35], it can be shown that a simple gradient ascent method can find the optimal maximizer with a sublinear rate. In other words, we can find a δ_Φ-accurate solution $\widehat{x}_{k,r}$ at the k^{th} iteration in finite steps. The accelerated gradient methods such as Nesterov acceleration can also be applied here to achieve a faster linear convergence rate.

In reality, there exist various delays such as forward delay T_f from the source to the target link and the backward delay T_b for certain feedback to the source. By choosing the stepsize inversely proportional to the maximum delay over the network, we enable to establish the asymptotic stability of the lower-level process (see Section 2.6 in [40]) as well as a nonasymptotic convergence guarantee (see [24]). Thus, as long as we execute a sufficiently long time for this lower-level process, we can obtain a desired δ_ϕ-accurate solution.

8 SIMULATION STUDIES

8.1 Validation of Bilevel Objective function

In this section, we conduct experiments to underpin Theorem 2 to demonstrate that our bilevel optimization based approach in Algorithm 1 recovers the standard network utility maximization solution with known utility functions. We consider a a single-link multi-user setting as in Section 5, where n users transmit their package in a single communication link with capacity P. We consider the following problem setup.

$$\max_{\boldsymbol{\alpha}\in\mathcal{A}}\ \Psi(\boldsymbol{\alpha})=\sum_{r=1}^n\frac{x_r^*(\boldsymbol{\alpha})^{1-\widetilde{\alpha}_r}}{1-\widetilde{\alpha}_r},$$

$$x_1^*(\boldsymbol{\alpha}),\ldots,x_n^*(\boldsymbol{\alpha})=\arg\max_{x_r>0,\sum_{r=1}^n x_r\le P}\sum_{r=1}^n\frac{x_r^{1-\alpha_r}}{1-\alpha_r}-B\Big(\sum_{i=1}^n x_i\Big)$$

where we choose the log barrier regularization function $B(x)=-\tau\log(P-x)$ with a parameter τ. For the lower-level problem, we

use a simple T-step gradient ascent method with stepsize λ to obtain good estimates $\widehat{x}_{k,r}, r = 1, ..., n$ at each round k. For the constraint set, we choose $\mathcal{A} := \{0.001 < a_r \leq 100, r = 1, ..., n\}$ to ensure the boundedness of $\boldsymbol{\alpha}$.

Hyperparameter selection. We choose the hyperparameters λ, η, β and τ from the candidate set $\{10^{-t}, t = -4, -3. -2, -1, 0, 1, 2, 3, 4\}$, and set a large inner-loop iteration number T from $\{10^t, t = 3, 4, 5\}$ to ensure a high-accuracy lower-level solution at each round. For all experiments, we choose the link capacity $P = 100$. For the experiment in Figure 2, we consider a 3-user setting with $n = 3$, where we set $\widetilde{\alpha}_1 = \frac{1}{2}, \widetilde{\alpha}_2 = \frac{2}{3}$ and $\widetilde{\alpha}_3 = \frac{2}{3}$. For the experiment in Figure 3, we consider a 5-user setting with $n = 3$, where we set $\widetilde{\alpha}_1 = \frac{1}{2}, \widetilde{\alpha}_2 = \frac{2}{5}$, $\widetilde{\alpha}_3 = \frac{3}{5}, \widetilde{\alpha}_4 = \frac{2}{3}, \widetilde{\alpha}_5 = \frac{2}{3}$.

Results. It can be seen from the left plot in Figure 2 that the total underlying utility achieved by our proposed DBiNUM increases with the number of rounds, and converges to the standard NUM solution 31.77. From the middle plot in in Figure 2 , it is shown that under the choice of $\widetilde{\alpha}_1, \widetilde{\alpha}_2, \widetilde{\alpha}_3 = \frac{1}{2}, \frac{2}{3}, \frac{2}{3}$ for the underlying utility functions, x_1 converges to the standard NUM solution 57.9, and x_2 and x_3 converge to the same solution 20.99 due to the identical underlying utility function with $\widetilde{\alpha}_2 = \widetilde{\alpha}_3 = \frac{2}{3}$. This validates our results in Theorem 2, where we show that the bilevel solutions $x_r^*(\boldsymbol{\alpha}^*), r = 1, ..., n$ recover the standard NUM solution. The same observation can be made for the 5-user case, where users 4, 5 converges to the lowest 9.9 due to the largest $\widetilde{\alpha}_4 = \widetilde{\alpha}_5 = \frac{2}{3}$, and user 2 converges to the largest 45.9 due to the smallest $\widetilde{\alpha}_2 = \frac{2}{5}$ (note that larger α means lower increase rate at larger x and hence a smaller allocated resource). From the right plots in Figure 2 and Figure 3, since the global solution $\boldsymbol{\alpha}$ is not unique, we plot the normalized solution $\alpha_i/\alpha_1, i = 2, 3, \cdots$. It can be clearly seen that each normalized solution converges after some rounds.

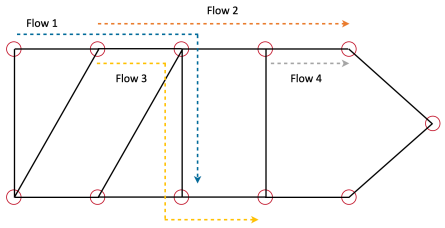

Figure 4: Abilene network with four transmission flows.

8.2 Simulation over Real-World Networks

In this section, we consider a real-world network, Abilene network, whose topology is shown in Figure 4. Following the setup in [7], this network contains four data transmission flows with distinct underlying utilities, where flow 1 has a quadratic utility $a_1 x^2$, flow 2 has a square root utility $a_2\sqrt{x + b_2} - a_2\sqrt{x}$, flow 3 has a log utility $a_3 \log(b_3 + 1)$, and flow 4 uses either an α-fairness $\frac{x^{1-a_4}}{1-a_4}$ or s-shape utility [43] $x^{a_4}\mathbf{1}_{(x\geq 0)} - b_4(-x)^{a_4}\mathbf{1}_{(x<0)}$ ($\mathbf{1}_{(\cdot)}$ is the indicator function). For the bilevel objective function in eq. (3), we choose a log barrier regularization function $B(x) = -\tau \log(P - x)$ with a capacity P. Similarly to the setup in Section 8.1, we use a simple T-step gradient ascent method with stepsize λ for the lower-level problem. For the constraint set, we choose $\mathcal{A} := \{1.01 < a_r \leq 100, r = 1, ..., n\}$ to ensure the boundedness of $\boldsymbol{\alpha}$.

Hyperparameter setting. For the regularization function $B(\cdot)$, we choose the constant $\tau = 0.01$ and set the capacity $P = 20$ for each link. The stepsizes λ, η, β are chosen from $\{10^t, t = -3, -2, -1, 0, 1, 2, 3\}$ to ensure the convergence. For the experiment in Figure 5, we set $a_1 = 0.1, a_2 = 5, b_2 = 0.4, a_3 = 4, b_3 = 1, a_4 = 0.8$ and $b_4 = 0.2$. For the experiment in Figure 6, we set $a_1 = 3, a_2 = 0.5, b_2 = 0.2, a_3 = 0.5, b_3 = 2, a_4 = 1.8$ and $b_4 = 2$.

Baseline. The baseline is the standard NUM solution, where each user has an α-fairness utility function with $\alpha = 2$.

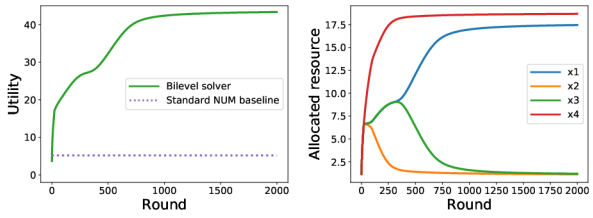

Figure 5: Performance of DBiNUM in Abilene network with $(a_1, a_2, b_2, a_3, b_3, a_4, b_4) = (0.1, 5, 0.4, 4, 1, 0.8, 0.2)$. Left: underlying utility Ψ v.s. # of rounds; right: resource v.s. # of rounds.

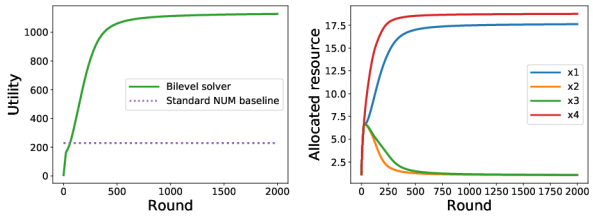

Figure 6: Performance of DBiNUM in Abilene network with $(a_1, a_2, b_2, a_3, b_3, a_4, b_4) = (3, 0.5, 0.2, 0.5, 2, 1.8, 2)$.

Results. It can be seen from the left plots in Figure 5 and Figure 5 that our bilevel optimization method iteratively increases the underlying network utility, and greatly outperform the standard NUM baseline. For example, in Figure 5, our DBiNUM method converges to a utility of 1127.06, which is much higher than the baseline 229.40. The same improvement can be observed from Figure 5. This demonstrate the effectiveness of our bilevel optimization process in increasing the total underlying network utility.

9 PROOF OF THEOREM 2

Let us first compute the lower-level solution $x_r^*(\boldsymbol{\alpha})$ of eq. (16). Based on the the standard analysis in [40], it is shown that the solutions satisfy the equality that $\sum_{r=1}^n x_r = P$, which implies that, for any index m, $x_m = P - \sum_{i\neq m} x_i$. Then, the solutions can be obtained by setting the derive of the objective w.r.t. x_j ($j \neq m$) to be 0, as shown below.

$$\frac{\partial(\sum_{i\neq m} \frac{x_i^{1-\alpha_i}}{1-\alpha_i} + \frac{(P-\sum_{i\neq m} x_i)^{1-\alpha_m}}{1-\alpha_m})}{\partial x_j} = x_j^{-\alpha_j} - \big(P - \sum_{i\neq m} x_i\big)^{-\alpha_m} = 0,$$

which further yields $x_j^{-\alpha_j} = x_m^{-\alpha_m}$ for any $j \neq m$. Then, combining this relationship with the equality $\sum_{i=1}^n x_i = P$, we have $x_r^*(\boldsymbol{\alpha}), r = 1, ..., n$ satisfy

$$\sum_{i=1}^n x_i^*(\boldsymbol{\alpha}) = P, \quad x_1^*(\boldsymbol{\alpha})^{-\alpha_1} = \cdots = x_n^*(\boldsymbol{\alpha})^{-\alpha_n}. \tag{20}$$

Next, we derive the solutions of $\boldsymbol{\alpha}^*$ from eq. (16). From eq. (20),

$$\sum_{t=1}^n (x_i^*)^{\frac{\alpha_i}{\alpha_t}} = P, \tag{21}$$

where we omit $\boldsymbol{\alpha}$ for each x_i^* to simplify notations. Then, for $i \neq j$, we derive the derivative $\frac{\partial x_i^*}{\partial \alpha_j}$ through setting the derivative of eq. (21) w.r.t. α_j to be 0 via implicit differentiation, as shown below.

$$\sum_{t=1}^n \frac{\alpha_i}{\alpha_t}(x_i^*)^{\frac{\alpha_i}{\alpha_t}-1}\frac{\partial x_i^*}{\partial \alpha_j} - (x_i^*)^{\frac{\alpha_i}{\alpha_j}}\frac{\alpha_i}{\alpha_j^2}\ln x_i^* = 0,$$

which, by rearranging all terms, yields

$$\frac{\partial x_i^*}{\partial \alpha_j} = \frac{(x_i^*)^{\frac{\alpha_i}{\alpha_j}}\frac{\alpha_i}{\alpha_j^2}\ln x_i^*}{\sum_{t=1}^n \frac{\alpha_i}{\alpha_t}(x_i^*)^{\frac{\alpha_i}{\alpha_t}-1}}. \tag{22}$$

For the case when $i = j$, using an approach similar to eq. (22), we have

$$\frac{\partial x_i^*}{\partial \alpha_i} = \frac{-\sum_{t\neq i}\frac{1}{\alpha_t}(x_i^*)^{\frac{\alpha_i}{\alpha_t}}\ln x_i^*}{\sum_{t=1}^n \frac{\alpha_i}{\alpha_t}(x_i^*)^{\frac{\alpha_i}{\alpha_t}-1}}. \tag{23}$$

Based on the property of the derivatives we obtain in eq. (22) and eq. (23), we next derive the optimal solution $\boldsymbol{\alpha}^*$ and the resulting resource allocation $x_i^*(\boldsymbol{\alpha}^*)$. Taking the derivative of the total upper-level objective $\Psi(\boldsymbol{\alpha}) = \sum_{i=1}^n \frac{x_r^*(\boldsymbol{\alpha})^{1-\widetilde{\alpha}_r}}{1-\widetilde{\alpha}_r}$ w.r.t. α_j yields

$$\frac{\partial \Psi(\boldsymbol{\alpha})}{\partial \alpha_j} = \sum_{i\neq j}(x_i^*)^{-\widetilde{\alpha}_i}\frac{\partial x_i^*}{\partial \alpha_j} + (x_j^*)^{-\widetilde{\alpha}_j}\frac{\partial x_j^*}{\partial \alpha_j},$$

which, combined with eq. (22) and eq. (23), yields

$$\begin{aligned}\frac{\partial \Psi(\boldsymbol{\alpha})}{\partial \alpha_j} =& \sum_{i\neq j}(x_i^*)^{-\widetilde{\alpha}_i}\frac{(x_i^*)^{\frac{\alpha_i}{\alpha_j}}\frac{\alpha_i}{\alpha_j^2}\ln x_i^*}{\sum_{t=1}^n \frac{\alpha_i}{\alpha_t}(x_i^*)^{\frac{\alpha_i}{\alpha_t}-1}} \\ &- (x_j^*)^{-\widetilde{\alpha}_j}\frac{\sum_{i\neq j}\frac{1}{\alpha_i}(x_j^*)^{\frac{\alpha_j}{\alpha_i}}\ln x_j^*}{\sum_{t=1}^n \frac{\alpha_j}{\alpha_t}(x_j^*)^{\frac{\alpha_j}{\alpha_t}-1}}.\end{aligned} \tag{24}$$

For the right hand side of eq. (24), note that

$$\begin{aligned}&\frac{\frac{1}{\alpha_i}(x_j^*)^{\frac{\alpha_j}{\alpha_i}}\ln x_j^*}{\sum_{t=1}^n \frac{\alpha_j}{\alpha_t}(x_j^*)^{\frac{\alpha_j}{\alpha_t}-1}} \overset{(i)}{=} \frac{\frac{1}{\alpha_j}x_j^*\ln x_j^*}{\sum_{t=1}^n \frac{\alpha_i}{\alpha_t}(x_j^*)^{\alpha_j(\frac{1}{\alpha_t}-\frac{1}{\alpha_i})}} \\ \overset{(ii)}{=}& \frac{\frac{1}{\alpha_j}x_j^*\ln x_j^*}{\sum_{t=1}^n \frac{\alpha_i}{\alpha_t}(x_i^*)^{\frac{\alpha_i}{\alpha_t}-1}} \overset{(iii)}{=} \frac{\frac{\alpha_i}{\alpha_j^2}(x_i^*)^{\frac{\alpha_i}{\alpha_j}}\ln x_i^*}{\sum_{t=1}^n \frac{\alpha_i}{\alpha_t}(x_i^*)^{\frac{\alpha_i}{\alpha_t}-1}},\end{aligned} \tag{25}$$

where (i) follows by dividing upper and lower sides by $\frac{\alpha_j}{\alpha_i}(x_j^*)^{\frac{\alpha_j}{\alpha_i}-1}$, (ii) follows because $(x_j^*)^{\alpha_j} = (x_i^*)^{\alpha_i}$ (see eq. (20)), and (iii) follows because $x_j^* = (x_i^*)^{\frac{\alpha_i}{\alpha_j}}$. Then, incorporate eq. (25) into eq. (24) yields

$$\frac{\partial \Psi(\boldsymbol{\alpha})}{\partial \alpha_j} = \frac{1}{\alpha_j}\ln x_j^* \sum_{i\neq j}((x_i^*)^{-\widetilde{\alpha}_i} - (x_j^*)^{-\widetilde{\alpha}_j})\frac{(x_i^*)^{\frac{\alpha_i}{\alpha_j}}}{\sum_{t=1}^n \frac{\alpha_i}{\alpha_t}(x_i^*)^{\frac{\alpha_i}{\alpha_t}-1}}. \tag{26}$$

We next consider two cases $P \neq n$ and $P = n$, separately.

For $P \neq n$ case: Note that $x_j^* \neq 1$. Otherwise, from eq. (21), we have $n = P$, which contradicts the condition that $P \neq n$. Then, we derive the optimal solution $\boldsymbol{\alpha}$ by setting eq. (26) to be 0. This gives

$$\sum_{i\neq j}((x_i^*)^{-\widetilde{\alpha}_i} - (x_j^*)^{-\widetilde{\alpha}_j})\frac{(x_i^*)^{\frac{\alpha_i^*}{\alpha_j^*}}}{\sum_{t=1}^n \frac{\alpha_i^*}{\alpha_t^*}(x_i^*)^{\frac{\alpha_i^*}{\alpha_t^*}-1}} = 0 \tag{27}$$

Let j be such that $(x_j^*)^{-\widetilde{\alpha}_j} \leq (x_i^*)^{-\widetilde{\alpha}_i}$ for any $i = 1, ..., n$, which, combined with eq. (27) and $x_i^* > 0$, yields

$$(x_1^*)^{-\widetilde{\alpha}_1} = \cdots = (x_n^*)^{-\widetilde{\alpha}_n}. \tag{28}$$

Combining the above eq. (28) with the relationship $(x_1^*)^{-\alpha_1} = \cdots = (x_n^*)^{-\alpha_n}$ in eq. (20) and the constraint $\boldsymbol{\alpha} \in \mathcal{A}$, the solution $\boldsymbol{\alpha}^* \in \arg\max_{\boldsymbol{\alpha}\in\mathcal{A}} \Psi(\boldsymbol{\alpha})$ satisfies that $\alpha_r^* = c\widetilde{\alpha}_r$ with $0 < c \leq \frac{b}{\max_r(\widetilde{\alpha}_r)}$ for $r = 1, ..., n$. Next, we show that the resulting $\boldsymbol{x}^*(\boldsymbol{\alpha}^*)$ recovers the solution of the conventional network maximization problem in eq. (17). To see this, combining eq. (28) with the relationship $\sum_{i=1}^n x_i^* = P$ in eq. (20) yields that each x_i^* satisfies $\sum_{t=1}^n (x_i^*)^{\frac{\widetilde{\alpha}_i}{\widetilde{\alpha}_t}} = P$, which can be verified to be the solution of eq. (17).

For $P = n$ case: In this case, letting the derivative in eq. (26) to be 0, it can be seen that if there exists at least one $x_j^* = 1$, based on eq. (20), all $x_1^*, ..., x_n^*$ equal to 1. Otherwise, using an approach similar to the above $P \neq n$ case, we have $\sum_{t=1}^n (x_i^*)^{\frac{\widetilde{\alpha}_i}{\widetilde{\alpha}_t}} = P$. Let i_0 be such that $\widetilde{\alpha}_{i_0} := \max_i \widetilde{\alpha}_i$. Then, the equation $\sum_{t=1}^n (x_{i_0}^*)^{\frac{\widetilde{\alpha}_{i_0}}{\widetilde{\alpha}_t}} = P = n$ implies that $x_{i_0}^* = 1$. Then, by eq. (20), we also have the conclusion that all $x_1^*, ..., x_n^*$ equal to 1. In sum, in this $P = n$ case, we have the solution given by $x_1^* = \cdots = x_n^* = 1$. Note that this also recovers the solution of eq. (17) in the case of $P = n$. Then, combining the above two cases completes the proof.

10 CONCLUSION AND FUTURE WORK

We provide a novel distributed bilevel approach for network utility maximization with unknown user utility functions. Our method iteratively improves the underlying total utility based on the user feedback. Theoretically, we analyze the convergence rate of the proposed method, and show that it also recovers the standard solutions when the utility functions are known. We anticipate that our proposed theory and algorithms can motivate the design of feasible resource allocation protocol to support distributed AI in dynamic, heterogeneous wireless networks. We anticipate that our results will promote the development of distributed bilevel optimization in the resource allocation over the communication networks.

ACKNOWLEDGMENTS

The work of Kaiyi Ji is supported in part by NSF under grants 2326592 and 2311274. The work of Lei Ying is supported in part by NSF under grants 2112471 and 2207548.

REFERENCES

[1] Amir Beck, Angelia Nedić, Asuman Ozdaglar, and Marc Teboulle. 2014. An O(1/k) gradient method for network resource allocation problems. *IEEE Transactions on Control of Network Systems* 1, 1 (2014), 64–73.

[2] Xuxing Chen, Minhui Huang, and Shiqian Ma. 2022. Decentralized bilevel optimization. *arXiv preprint arXiv:2206.05670* (2022).

[3] M. Chiang. 2005. Balancing Transport and Physical Layers in Wireless Multihop Networks: Jointly Optimal Congestion Control and Power Control. *IEEE Journal of Selected Areas in Communications* 23(1) (Jan 2005), 104–116.

[4] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle. 2007. Layering as optimization decomposition: A mathematical theory of network architectures. In *Proceedings of the IEEE*. 255–312.

[5] Justin Domke. 2012. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics (AISTATS)*. PMLR, 318–326.

[6] A. Eryilmaz and R. Srikant. 2004. Scheduling with quality of service constraints over Rayleigh fading channels. In *Proc. IEEE Conf. Decision and Control (CDC)*, Vol. 4. 3447–3452.

[7] Xinzhe Fu and Eytan Modiano. 2022. Learning-NUM: Network utility maximization with unknown utility functions and queueing delay. *IEEE/ACM Transactions on Networking* (2022).

[8] Hongchang Gao, Bin Gu, and My T Thai. 2022. Stochastic bilevel distributed optimization over a network. *arXiv preprint arXiv:2206.15025* (2022).

[9] Leonidas Georgiadis, Michael J Neely, Leandros Tassiulas, et al. 2006. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends® in Networking* 1, 1 (2006), 1–144.

[10] Javad Ghaderi, Sanjay Shakkottai, and Rayadurgam Srikant. 2016. Scheduling storms and streams in the cloud. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)* 1, 4 (2016), 1–28.

[11] Saeed Ghadimi and Mengdi Wang. 2018. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246* (2018).

[12] Xiaowen Gong, Xu Chen, and Junshan Zhang. 2014. Social group utility maximization in mobile networks: From altruistic to malicious behavior. In *2014 48th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 1–6.

[13] Mohammad H Hajiesmaili, Ahmad Khonsari, Ali Sehati, and Mohammad Sadegh Talebi. 2012. Content-aware rate allocation for efficient video streaming via dynamic network utility maximization. *Journal of Network and Computer Applications* 35, 6 (2012), 2016–2027.

[14] Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. 2020. A Two-Timescale Framework for Bilevel Optimization: Complexity Analysis and Application to Actor-Critic. *arXiv preprint arXiv:2007.05170* (2020).

[15] Kaiyi Ji. 2021. Bilevel Optimization for Machine Learning: Algorithm Design and Convergence Analysis. *arXiv preprint arXiv:2108.00330* (2021).

[16] Kaiyi Ji and Yingbin Liang. 2021. Lower Bounds and Accelerated Algorithms for Bilevel Optimization. *arXiv preprint arXiv:2102.03926* (2021).

[17] Kaiyi Ji, Mingrui Liu, Yingbin Liang, and Lei Ying. 2022. Will Bilevel Optimizers Benefit from Loops. *arXiv preprint arXiv:2205.14224* (2022).

[18] Kaiyi Ji, Guocong Quan, and Jian Tan. 2018. Asymptotic miss ratio of LRU caching with consistent hashing. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 450–458.

[19] Kaiyi Ji, Junjie Yang, and Yingbin Liang. 2021. Bilevel optimization: Convergence analysis and enhanced design. In *International Conference on Machine Learning (ICML)*. PMLR, 4882–4892.

[20] Kaiyi Ji and Lei Ying. 2023. Network Utility Maximization with Unknown Utility Functions: A Distributed, Data-Driven Bilevel Optimization Approach. *arXiv preprint arXiv:2301.01801* (2023).

[21] Frank P Kelly, Aman K Maulloo, and David Kim Hong Tan. 1998. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society* 49, 3 (1998), 237–252.

[22] J-W Lee, Ravi R Mazumdar, and Ness B Shroff. 2005. Non-convex optimization and rate control for multi-class services in the Internet. *IEEE/ACM transactions on networking* 13, 4 (2005), 827–840.

[23] Chengtie Li, Jinkuan Wang, and Mingwei Li. 2016. Data transmission optimization algorithm for network utility maximization in wireless sensor networks. *International Journal of Distributed Sensor Networks* 12, 9 (2016), 1550147716670646.

[24] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. 2017. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. *Advances in Neural Information Processing Systems (NeurIPS)* 30 (2017).

[25] Xiaojun Lin and Ness B Shroff. 2006. Utility maximization for communication networks with multipath routing. *IEEE Trans. Automat. Control* 51, 5 (2006), 766–781.

[26] Risheng Liu, Jiaxin Gao, Jin Zhang, Deyu Meng, and Zhouchen Lin. 2021. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[27] Risheng Liu, Pan Mu, Xiaoming Yuan, Shangzhi Zeng, and Jin Zhang. 2020. A generic first-order algorithmic framework for bi-level programming beyond lower-level singleton. In *International Conference on Machine Learning (ICML)*. PMLR, 6305–6315.

[28] Steven H Low and David E Lapsley. 1999. Optimization flow control. I. Basic algorithm and convergence. *IEEE/ACM Transactions on networking* 7, 6 (1999), 861–874.

[29] Songtao Lu, Xiaodong Cui, Mark S Squillante, Brian Kingsbury, and Lior Horesh. 2022. Decentralized Bilevel Optimization for Personalized Client Learning. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5543–5547.

[30] Yu Ma, Weifa Liang, and Wenzheng Xu. 2018. Charging utility maximization in wireless rechargeable sensor networks by charging multiple sensors simultaneously. *IEEE/ACM Transactions on Networking* 26, 4 (2018), 1591–1604.

[31] Siva Theja Maguluri, Rayadurgam Srikant, and Lei Ying. 2012. Stochastic models of load balancing and scheduling in cloud computing clusters. In *IEEE INFOCOM 2012*. IEEE, 702–710.

[32] J. Mo and J. Walrand. 2000. Fair end-to-end window-based congestion control. *IEEE/ACM Trans. Netw.* 5 (2000), 556–567.

[33] Michael J Neely, Eytan Modiano, and Chih-Ping Li. 2008. Fairness and optimal stochastic control for heterogeneous networks. *IEEE/ACM Transactions On Networking* 16, 2 (2008), 396–409.

[34] Michael J Neely, Eytan Modiano, and Charles E Rohrs. 2003. Dynamic power allocation and routing for time varying wireless networks. In *IEEE INFOCOM 2003.*, Vol. 1. IEEE, 745–755.

[35] Yurii Nesterov. 2018. Smooth convex optimization. In *Lectures on convex optimization*. Springer, 59–137.

[36] Yurii Nesterov and Vladimir Spokoiny. 2017. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics* 17, 2 (2017), 527–566.

[37] Daniel P Palomar and Mung Chiang. 2007. Alternative distributed algorithms for network utility maximization: Framework and applications. *IEEE Trans. Automat. Control* 52, 12 (2007), 2254–2269.

[38] Fabian Pedregosa. 2016. Hyperparameter optimization with approximate gradient. In *International conference on machine learning*. PMLR, 737–746.

[39] Peiwen Qiu, Yining Li, Zhuqing Liu, Prashant Khanduri, Jia Liu, Ness B Shroff, Elizabeth Serena Bentley, and Kurt Turck. 2022. DIAMOND: Taming Sample and Communication Complexities in Decentralized Bilevel Optimization. *arXiv preprint arXiv:2212.02376* (2022).

[40] Rayadurgam Srikant and Lei Ying. 2013. *Communication networks: an optimization, control, and stochastic networks perspective*. Cambridge University Press.

[41] A. L. Stolyar. 2005. Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm. *Queueing Syst.* 50, 4 (August 2005), 401–457.

[42] Mohammad Sadegh Talebi, Ahmad Khonsari, Mohammad Hassan Hajiesmaili, and Sina Jafarpour. 2011. Quasi-optimal network utility maximization for scalable video streaming. *arXiv preprint arXiv:1102.2604* (2011).

[43] Amos Tversky and Daniel Kahneman. 1992. Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and uncertainty* 5, 4 (1992), 297–323.

[44] Arun Verma and Manjesh K Hanawal. 2020. Stochastic network utility maximization with unknown utilities: Multi-armed bandits approach. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 189–198.

[45] X. Wang and K. Kar. 2005. Cross-layer rate control for end-to-end proportional fairness in wireless networks with random access. In *Proc. ACM Int. Symp. Mobile Ad Hoc Networking and Computing (MobiHoc)*. 157–168.

[46] Pradeep Chathuranga Weeraddana, Marian Codreanu, Matti Latva-aho, and Anthony Ephremides. 2011. Resource allocation for cross-layer utility maximization in wireless networks. *IEEE Transactions on Vehicular Technology* 60, 6 (2011), 2790–2809.

[47] Ermin Wei, Asuman Ozdaglar, and Ali Jadbabaie. 2013. A distributed Newton method for network utility maximization–I: Algorithm. *IEEE Trans. Automat. Control* 58, 9 (2013), 2162–2175.

[48] Shuoguang Yang, Xuezhou Zhang, and Mengdi Wang. 2022. Decentralized gossip-based stochastic bilevel optimization over communication networks. *arXiv preprint arXiv:2206.10870* (2022).

[49] Xiaoguo Ye and Weifa Liang. 2017. Charging utility maximization in wireless rechargeable sensor networks. *Wireless Networks* 23, 7 (2017), 2069–2081.

[50] L. Ying, R. Srikant, A. Eryilmaz, and G. Dullerud. 2007. Distributed Fair Resource Allocation in Cellular Networks in the Presence of Heterogeneous Delays. *IEEE Trans. Autom. Control* 52, 1 (January 2007), 129–134.