Distributed Optimization for Network Resource Allocation with Nonsmooth Utility Functions

Hideaki Iiduka

***Abstract*—The network utility maximization (NUM) problem is the problem of maximizing the overall utility of a network under capacity constraints, where each source in the network has its own private nonsmooth concave utility function (which allows the true utility to be modeled accurately) and each link in the network has only its capacity constraint. To solve this problem, two distributed optimization algorithms are proposed: a projected proximal algorithm and a projected subgradient algorithm. These algorithms can be implemented for the case that each source tries to maximize only its utility by using its proximity operator or subdifferential and each link tries to satisfy only its capacity constraint by using the metric projection onto its capacity constraint set. A convergence analysis indicates that these algorithms are sufficient for each source to find the optimal resource allocation. The convergence, optimality, and performance of the proposed algorithms are demonstrated through numerical comparisons with the existing decentralized network flow control algorithm.**

***Index Terms*—Distributed optimization, metric projection, network utility maximization, nonsmooth utility function, proximity operator, subdifferential.**

I. Introduction

In modern communication networks, a critical problem is how best to allocate the network resources. One method of addressing this problem is to treat it as a utility-based resource allocation problem, for which the goal is to determine the source rates that maximize the overall utility, that is, the utility aggregated over all sources, where the constraints are all of the link capacity constraints. This problem is referred to as the *network utility maximization* (NUM) problem [13], [26, Chapter 2].

In the case that all sources have the same logarithmic utility function, the resource allocation from the NUM is said to be *proportionally fair* [13], [20], [26, Chapter 2] (refer to, e.g., [20] for a comparison of standard definitions of fairness). Various optimization algorithms for solving the NUM problem have been reported that can be applied to the standard smooth concave functions. For example, asynchronous distributed algorithms [17] use projected gradient algorithms to perform NUM. Wei et al. [27], [28] proposed a distributed Newton-type fast converging algorithm for the NUM problem that has a superlinear rate of convergence. Beck et al. [2] proposed a fast distributed gradient method as an alternative to decentralized algorithms using dual-based gradient methods. Marašević et al. [18] proposed a fast distributed stateless algorithm for general smooth utility maximization problems. For cases that the utility functions are differentiable but nonconcave, various iterative algorithms [10], [12] have been proposed.

When each source has a transmission rate demand for some application's service, the utility of the source increases with increasing transmission rate for transmission rates less than the demand and is constant for transmission rates above the demand. Thus, the utility function of the source is a nonsmooth concave function[1]. Therefore, in the present work on the NUM problem, we assume that the utility function of each source is concave but *not necessarily* differentiable at all points.

In the case of nonsmooth concave utility functions, typical techniques using the (Lipschitz continuous) gradients and Hessian of the utility functions are not available for the NUM problem. Therefore, for the present work, we use *proximity operators* [1, Definition 12.23], [19], [21], [25] of nonsmooth concave utility functions, which are utilized in practical problems in signal processing [7], [8]. We also use *subdifferentials* [1, Definition 16.1] of nonsmooth concave utility functions, which are useful for solving distributed convex optimization problems [3], [22]. Using a proximity operator or a subdifferential, each source can separately maximize its own utility[2].

There have been useful algorithms that can be applied to the NUM problem with nonsmooth concave utility functions. For example, a distributed algorithm [25] uses proximity operators of utility functions and the metric projection using all of the link capacity constraints. Bertsekas [3] proposed incremental optimization algorithms using the metric projection onto the intersection of all of the link capacity constraint sets. Nedić and Ozdaglar [23], [24] proposed dual subgradient and primal-dual subgradient methods to solve the NUM problem in a distributed manner. Recently, Yu and Neely [30] proposed a fast dual subgradient method that can be applied to the case where utility functions are concave.

The main objective of this paper is to propose distributed optimization algorithms that enable each source to find the optimal solution to the NUM problem for the case that each source tries to maximize its utility and each link tries to satisfy only its own capacity constraint. To address this goal, we consider a network system [17], [30] such that each source can communicate with links that it uses and each link can communicate with sources that uses it. We refer to sources

H. Iiduka is with the Department of Computer Science, Meiji University, Kanagawa 214-8571, Japan (e-mail: iiduka@cs.meiji.ac.jp). This work was supported by JSPS KAKENHI Grant Numbers JP15K04763 and JP18K11184.

[1]As a simple example showing that a utility function need not be differentiable, suppose that the transmission rate of source s is x_s and that the transmission rate demand is r_0. Then the utility function defined as $u_s(x_s) = x_s$ for $x_s \leq r_0$ and $u_s(x_s) = r_0$ for $x_s > r_0$ is nondifferentiable.

[2]The computational efficient of the proximity operators of several commonly used functions, such as the l_1-norm, quadratic polynomial, logarithm, and sums of these functions, are shown in Tables 10.1 and 10.2 in [8]. Examples of functions of which subdifferentials can be efficiently computed are shown in [1, Chapters 16–18].

and links in the communication network as *users* and assume that each user and its neighboring users that can communicate with it form a subnetwork in which each user can transmit its estimate to its neighboring users.

For this research, we propose two distributed optimization algorithms and apply these algorithms to the solution of the NUM problem. The first proposed algorithm is a projected proximal algorithm based on the random projected proximal algorithm [11, Algorithm 3.1]. The second proposed algorithm is a projected subgradient algorithm based on the random projected subgradient algorithm [11, Algorithm 4.1]. The proposed algorithms can work for the case where, at each time, each user receives its neighboring users' estimates, computes their average, and updates its estimate by using the computable metric projection and proximal point (or subgradient) of its own private utility function at the average.

The proposed algorithms are related to the existing algorithms for solving the NUM problem with nonsmooth concave utility functions described in the fifth paragraph of this section. The algorithms in [3], [25] need to use the metric projection onto the intersection of all the link capacity constraint sets (see algorithms (8) and (9) for the details of the algorithms in [3], [25]). Since this intersection is complicated, it would be difficult to compute the metric projection. The proposed algorithms can be implemented using only the computable metric projections onto half-spaces and can solve the NUM problem over the complicated intersection of all the capacity constraint sets. The dual subgradient and primal-dual subgradient methods [23], [24] can be implemented under the assumption that there exist "network providers" [23, p.1761] knowing all of the link capacities (see algorithm (11) for the details of the method in [23]). However, it might not be realistic to assume the existences of network providers. The proposed algorithms can be implemented under the assumption that there is no user that knows all of the link capacities and that each link knows only its own private link capacity. The decentralized network flow control (DNFC) algorithm [30, Algorithm 2] can work for the case where each user transmits its estimate to its neighboring users and can solve the NUM problem considered in the present paper [30, Theorem 3, (23)] (see algorithm (13) for the details of the DNFC algorithm).

This paper makes the following three contributions. First, this paper formulates the NUM problem as a *nonsmooth* concave optimization problem over the intersection of closed convex sets (Problem II.1). Allowing the utility function of each source to be nonsmooth enables it to accurately model the true utility. Secondly, our convergence analysis indicates that the whole sequence generated by either of our two proposed algorithms with diminishing step sizes converges to the NUM problem solution (Theorems III.1 and III.2). In particular, the present paper presents a convergence rate analysis of the proposed algorithms (Theorem III.1(ii)) that could not be performed using the analysis method in [11, Subsections 3.2 and 4.2] of random projection algorithms. We also perform a detailed convergence analysis of the proposed algorithms with constant step sizes (Theorem III.1(i)), unlike [11]. Finally, the behaviors of the two proposed algorithms are compared to that of the DNFC algorithm [30, Algorithm 2] and it is shown that the proposed algorithms with diminishing step sizes performed better than the DNFC algorithm.

The remainder of this paper is organized as follows. Section II gives the constraint sets and utility functions of the users and states the NUM problem in terms of a nonsmooth concave optimization over the intersection of closed convex sets. Section III gives the details of the proposed projected proximal and subgradient algorithms and shows the results of a convergence analysis. It also provides comparisons of the proposed algorithms with the existing algorithms for the NUM problem. In Section IV, a concrete NUM problem is examined and the behaviors of the two proposed algorithms and the DNFC algorithm are compared numerically. First, however, we present the notation and some definitions used herein.

A. Notation and Definitions

$\mathbb{R}^S$ denotes an S-dimensional Euclidean space with inner product $\langle\cdot,\cdot\rangle$ which induces norm $\|\cdot\|$, and $\mathbb{R}^S_+ := \{\boldsymbol{x} := (x_s)_{s=1}^S \in \mathbb{R}^S \colon x_s \geq 0 \ (s = 1, 2, \ldots, S)\}$. $\mathbb{N}$ denotes the set of natural numbers (namely, the positive integers and zero). $[W]_{ij}$ and $W^\top$ denote the (i,j)th entry and the transpose of a matrix W. Id denotes the identity mapping on $\mathbb{R}^S$. The *fixed point set* of a mapping $T \colon \mathbb{R}^S \to \mathbb{R}^S$ is denoted by $\mathrm{Fix}(T) := \{\boldsymbol{x} \in \mathbb{R}^S \colon T(\boldsymbol{x}) = \boldsymbol{x}\}$.

A mapping $T \colon \mathbb{R}^S \to \mathbb{R}^S$ is *nonexpansive* [1, Definition 4.1(ii)] if $\|T(\boldsymbol{x}) - T(\boldsymbol{y})\| \leq \|\boldsymbol{x} - \boldsymbol{y}\|$ $(\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^S)$, and T is *firmly nonexpansive* if $\|T(\boldsymbol{x}) - T(\boldsymbol{y})\|^2 + \|(\mathrm{Id} - T)(\boldsymbol{x}) - (\mathrm{Id} - T)(\boldsymbol{y})\|^2 \leq \|\boldsymbol{x} - \boldsymbol{y}\|^2$ $(\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^S)$. The metric projection onto a nonempty closed convex set $C \subset \mathbb{R}^S$ is denoted by P_C, and $P_C(\boldsymbol{x})$ $(\boldsymbol{x} \in \mathbb{R}^S)$ is such that

$$P_C(\boldsymbol{x}) \in C \text{ and } \|\boldsymbol{x} - P_C(\boldsymbol{x})\| = \mathrm{d}(\boldsymbol{x}, C) := \inf_{\boldsymbol{y} \in C} \|\boldsymbol{x} - \boldsymbol{y}\|.$$

The mapping P_C is firmly nonexpansive and satisfies $\mathrm{Fix}(P_C) = C$ [1, Proposition 4.8, (4.8)].

Given a convex function $f \colon \mathbb{R}^S \to \mathbb{R}$, the *proximity operator* of f [1, Definition 12.23], [19], [21], denoted by Prox_f, maps every $\boldsymbol{x} \in \mathbb{R}^S$ to the unique minimizer of $f(\cdot) + (1/2)\|\boldsymbol{x} - \cdot\|^2$. That is, for all $\boldsymbol{x} \in \mathbb{R}^S$,

$$\{\mathrm{Prox}_f(\boldsymbol{x})\} = \operatorname*{argmin}_{\boldsymbol{y} \in \mathbb{R}^S} \left[f(\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 \right].$$

The *subdifferential* [1, Definition 16.1] of $f \colon \mathbb{R}^S \to \mathbb{R}$ is the set-valued operator defined for all $\boldsymbol{x} \in \mathbb{R}^S$ by

$$\partial f(\boldsymbol{x}) := \left\{ \boldsymbol{u} \in \mathbb{R}^S \colon f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \boldsymbol{y} - \boldsymbol{x}, \boldsymbol{u} \rangle \ \ \left(\boldsymbol{y} \in \mathbb{R}^S\right) \right\}.$$

A point $\boldsymbol{u} \in \partial f(\boldsymbol{x})$ is said to be a *subgradient* of f at $\boldsymbol{x}$.

A directed graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ is a finite nonempty set $\mathcal{V}$ of nodes (users) and a collection $\mathcal{E}$ of ordered pairs of distinct nodes from $\mathcal{V}$ [4, p. 394]. A directed graph is said to be *strongly connected* if, for each pair of nodes i and l, there exists a directed path from i to l [4, p. 394]. The network topology at time k is expressed as a directed graph $\mathcal{G}(k) := (\mathcal{V}, \mathcal{E}(k))$, where $\mathcal{E}(k) \subset \mathcal{V} \times \mathcal{V}$ and $(i,j) \in \mathcal{E}(k)$ stands for a link such that user i receives information from user j at time k. Let $\mathcal{N}_i(k) \subset \mathcal{V}$ be the set of users that send information to user i; i.e., $\mathcal{N}_i(k) := \{j \in \mathcal{V} \colon (i,j) \in \mathcal{E}(k)\}$ and $i \in \mathcal{N}_i(k)$ $(i \in \mathcal{V}, k \geq 0)$. We call user j $(j \in \mathcal{N}_i(k))$ *neighboring user* j of user i at time k.

II. System Model and Problem Formulation

We consider an abstract network comprising a set of sources $\mathcal{S} := \{1, 2, \dots, S\}$ and a set of links $\mathcal{L} := \{1, 2, \dots, L\}$, where link l has capacity $c_l \in \mathbb{R}_+$. Let $\mathcal{S}(l) \subset \mathcal{S}$ denote the set of sources that use link l and let $\mathcal{L}(s) \subset \mathcal{L}$ denote the set of links used by source s. The transmission rate of source s is denoted by $x_s \in \mathbb{R}_+$. We refer to $i \in \mathcal{V} := \mathcal{S} \cup \mathcal{L}$ as user i.

A. Capacity Constraints

The capacity constraint for link l is the restriction that the sum of the transmission rates of all the sources sharing the link be less than or equal to c_l. Accordingly, the capacity constraint set $C_l \subset \mathbb{R}^S$ of link l is defined by

$$C_l := \left\{ \boldsymbol{x} := (x_s)_{s\in\mathcal{S}} \in \mathbb{R}^S \colon \sum_{s\in\mathcal{S}(l)} x_s \le c_l \right\}. \tag{1}$$

We assume that link l has a constraint set C_l having a closed-form expression. This implies that link l knows the elements in $\mathcal{S}(l)$ and the value of c_l. Other users cannot know the information of link l. In this paper, we call the information that only user i knows user i's *private information*. Since C_l is a half-space, the metric projection P_{C_l} onto C_l can be computed within a finite number of arithmetic operations [1, Example 28.16][3]. Hence, $P_l := P_{C_l}$ is link l's private information.

We also assume that source s maximizes its utility over the interval $I_s := [0, M_s]$, where $M_s > 0$ denotes the maximum allowed rate for source s (see Subsection II-B for the definitions of utility functions). In this case, the constraint set of source s is defined by

$$C_s := \left\{ \boldsymbol{x} := (x_s)_{s\in\mathcal{S}} \in \mathbb{R}^S \colon x_s \in I_s := [0, M_s] \right\}. \tag{2}$$

Since no one knows the value of M_s except source s, the computable projection $P_s := P_{C_s}$ is source s's private information.

Consistent with the above discussion, the following is assumed.

Assumption II.1

(i) *Link l ($l \in \mathcal{L}$) has the metric projection P_l onto C_l with $\mathrm{Fix}(P_l) = C_l$, which is its own private information, where C_l is defined by* (1).

(ii) *Source s ($s \in \mathcal{S}$) has the metric projection P_s onto C_s with $\mathrm{Fix}(P_s) = C_s$, which is its own private information, where C_s is defined by* (2).

The constraint set in the NUM problem is composed of the intervals for maximum allowed rates for sources and the capacity constraints for links. Accordingly, the constraint set is defined as the following subset of $\mathbb{R}^S$:

$$\begin{aligned} C &:= (I_1 \times \cdots \times I_S) \cap \bigcap_{l\in\mathcal{L}} C_l = \bigcap_{s\in\mathcal{S}} C_s \cap \bigcap_{l\in\mathcal{L}} C_l \\ &= \bigcap_{s\in\mathcal{S}} \mathrm{Fix}(P_s) \cap \bigcap_{l\in\mathcal{L}} \mathrm{Fix}(P_l). \end{aligned} \tag{3}$$

[3]The metric projection P_H onto a half-space $H := \{\boldsymbol{x} \in \mathbb{R}^S \colon \langle \boldsymbol{a}, \boldsymbol{x}\rangle \le r\}$ is given by $P_H(\boldsymbol{x}) = \boldsymbol{x}$ for $\boldsymbol{x} \in H$ and $P_H(\boldsymbol{x}) = \boldsymbol{x} + ((r - \langle \boldsymbol{a}, \boldsymbol{x}\rangle)/\|\boldsymbol{a}\|^2)\boldsymbol{a}$ for $\boldsymbol{x} \notin H$, where $r \in \mathbb{R}$, $\boldsymbol{a} \neq 0$, and functions $\langle \cdot, \cdot \rangle$ and $\|\cdot\|$ are respectively the inner product and norm of $\mathbb{R}^S$.

B. Utility Function

We assume that each source s in the network acts so as to maximize its own utility. The utility of source s is defined by the value of a *utility function* u_s at current rate x_s. The resource allocation corresponding to the utility function of source s defined for all $\boldsymbol{x} := (x_s)_{s\in\mathcal{S}} \in \mathbb{R}^S_+$ by

$$u_s(\boldsymbol{x}) := w_s \log(x_s + p_s), \tag{4}$$

where $w_s, p_s > 0$ are source s's private information, is said to be *weighted proportionally fair* [13], [20], [26, Chapter 2.1]. The function defined by (4) is differentiable and strictly concave [1, Proposition 17.13(i), (iv)], which implies that the utility of each source increases for any increase in its transmission rate. Moreover, the function u_s defined by (4) is Lipschitz continuous on C_s (see (2) for the definition of the constraint set of source s).

We also assume the existence of a source having utility which increases with increasing transmission rate when its transmission rate is less than a certain value (e.g., the rate needed for an application's service) but does not always increase when its transmission rate is more than this value. Such a utility function is, for example, expressed as the following nonsmooth, Lipschitz continuous function: for all $\boldsymbol{x} := (x_s)_{s\in\mathcal{S}} \in \mathbb{R}^S_+$,

$$u_s(\boldsymbol{x}) := -\max\{-w_s(x_s - a_s) + b_s, b_s\}, \tag{5}$$

where $w_s > 0$ and $a_s, b_s \in \mathbb{R}$ are source s's private information (see also Section IV). The function u_s defined by (5) is concave [7, Lemma 10]. Meanwhile, we assume that each link l in the network does not have utility of its own since the NUM problem is to maximize the sum of utility functions of all sources.

As seen in (4) and (5), the closed-form expression of u_s is source s's private information. From the above discussion, the following is assumed.

Assumption II.2

(i) *The utility function $u_s \colon \mathbb{R}^S \to \mathbb{R}$ of source s ($s \in \mathcal{S}$) is concave and Lipschitz continuous, which is its own private information. In addition, each source s has an associated proximity operator and an associated subdifferential.*

(ii) *The utility function $u_l \colon \mathbb{R}^S \to \mathbb{R}$ of link l ($l \in \mathcal{L}$) is defined for all $\boldsymbol{x} \in \mathbb{R}^S$ by $u_l(\boldsymbol{x}) = 0$.*

This proximity operator of $-u_s$ defined by either (4) or (5) can be easily computed within a finite number of arithmetic operations [7, Lemma 10], [8, Tables 10.1 and 10.2]. The subgradients of $-u_s$ defined by either (4) or (5) can be computed efficiently [1, Chapter 16]. Assumption II.2(ii) implies that, for all $l \in \mathcal{L}$, all $\boldsymbol{x} \in \mathbb{R}^S$, and all $\alpha > 0$, $\mathrm{Prox}_{-\alpha u_l}(\boldsymbol{x}) = \boldsymbol{x}$ and $\partial(-u_l)(\boldsymbol{x}) = \{\boldsymbol{0}\}$.

C. NUM problem

The main objective of this paper is to solve the NUM problem,

$$\text{maximize } \sum_{s\in\mathcal{S}} u_s(\boldsymbol{x}) \text{ subject to } \boldsymbol{x} \in C := \bigcap_{s\in\mathcal{S}} C_s \cap \bigcap_{l\in\mathcal{L}} C_l,$$

where $u_s\colon \mathbb{R}_+^S \to \mathbb{R}$ satisfies Assumption II.2(i), and C_l and C_s are the subsets of $\mathbb{R}^S$ defined by (1) and (2), respectively. Assumption II.2(ii) and (3), together with $\mathcal{V} := \mathcal{S} \cup \mathcal{L}$, thus imply that the NUM problem considered here can be formulated as follows.

Problem II.1 *Under Assumptions II.1 and II.2,*

$$\text{maximize } U(\boldsymbol{x}) := \sum_{i\in\mathcal{V}} u_i(\boldsymbol{x})$$
$$\text{subject to } \boldsymbol{x} \in C := \bigcap_{i\in\mathcal{V}} \mathrm{Fix}(P_i).$$

The solution set of Problem II.1 is nonempty since $\bigcap_{s\in\mathcal{S}} C_s$ is bounded and $C \subset \bigcap_{s\in\mathcal{S}} C_s = \bigcap_{s\in\mathcal{S}} \mathrm{Fix}(P_s)$ [1, Corollary 8.31, Proposition 11.14].

III. Distributed Optimization Algorithms

Assumptions II.1 and II.2 ensure that source s uses only the closed-form expressions of u_s and C_s and link l uses only the closed-form expressions of u_l and C_l. This section proposes two distributed optimization algorithms that enable each user to solve Problem II.1 without using other users' private information.

A. Assumptions

To address the goal of this paper, the following is assumed.

Assumption III.1 [17, p.864]

(i) *Source s ($s \in \mathcal{S}$) can communicate with link l ($l \in \mathcal{L}(s)$), and link l ($l \in \mathcal{L}$) can communicate with source s ($s \in \mathcal{S}(l)$).*

(ii) *At time k, each user i ($i \in \mathcal{V}$) and its neighboring users j ($j \in \mathcal{N}_i(k)$) form a network in which each user can transmit its estimate to its neighboring users.*

To solve Problem II.1 in a distributed manner, the following assumptions are also needed.

Assumption III.2 [16, Assumption 4] *There exists $Q \geq 1$ such that the graph $(\mathcal{V}, \bigcup_{l=0}^{Q-1} \mathcal{E}(k+l))$ is strongly connected for all $k \in \mathbb{N}$.*

Assumption III.3 [16, Assumption 5] *For $k \in \mathbb{N}$, user i has a set of weight parameters $w_{ij}(k)$ ($j \in \mathcal{V}$) satisfying the following:*

(i) *$w_{ij}(k) \geq 0$ for all $j \in V$ and $w_{ij}(k) = 0$ when $j \notin \mathcal{N}_i(k)$;*

(ii) *There exists $w \in (0,1)$ such that $w_{ij}(k) \geq w$ for all $j \in \mathcal{N}_i(k)$.*

(iii) *$\sum_{j\in\mathcal{V}} w_{ij}(k) = 1$ for all $i \in \mathcal{V}$ and $\sum_{i\in\mathcal{V}} w_{ij}(k) = 1$ for all $j \in \mathcal{V}$.*

A relationship exists between Assumptions III.2 and III.3. The matrix $W(k)$ defined by $[W(k)]_{ij} := w_{ij}(k)$ ($k \in \mathbb{N}$) satisfying Assumption III.3(i)–(iii) is said to be *doubly stochastic* [16, Assumption 5]. Theorem 1 in [29] indicates that the spectral radius ρ of the doubly stochastic matrix $W(k)$ ($k \in \mathbb{N}$) is less than one if and only if, for all $k \in \mathbb{N}$,

$$\lim_{t\to+\infty} W(k)^t = \frac{\boldsymbol{e}\boldsymbol{e}^\top}{S+L}, \tag{6}$$

where $\boldsymbol{e} \in \mathbb{R}^{S+L}$ denotes a column vector in which all the entries are equal to one. Moreover, Theorem 3.2.1 in [6] and the remark in [29, Theorem 1] show that (6) means the network is strongly connected. Accordingly, Assumption III.2 holds if $\rho < 1$. For example, $\rho < 1$ holds when one is a simple eigenvalue of $W(k)$ and all other eigenvalues are strictly less than one in magnitude [29, p.67]. We need methods for setting $w_{ij}(k)$ ($i \in \mathcal{V}, j \in \mathcal{N}_i(k)$) that satisfy Assumption III.3. For example, there is a decentralized method that can be implemented under the condition that users pass along messages in cyclic order. Our GitHub repository (URL: https://github.com/iiduka-researches/misc/blob/master/misc/matrix.py) provides the implementation in Python.

B. Projected Proximal Algorithm

Algorithm 1 is a projected proximal algorithm for solving Problem II.1 under the assumptions in Sections II and III-A.

Algorithm 1 Projected proximal algorithm

Require: $(\alpha_k)_{k\in\mathbb{N}} \subset (0, +\infty)$

1: $k \leftarrow 0$, $\boldsymbol{x}_i(0) \in \mathbb{R}^S$ ($i \in \mathcal{V} := \mathcal{S} \cup \mathcal{L}$)
2: **loop**
3: **for** $i = 1$ to $i = V := S + L$ **do**
4: $\boldsymbol{v}_i(k) := \sum_{j\in\mathcal{N}_i(k)} w_{ij}(k)\boldsymbol{x}_j(k)$

$$\boldsymbol{x}_i(k+1) := P_i\left(\mathrm{Prox}_{-\alpha_k u_i}(\boldsymbol{v}_i(k))\right) = \begin{cases} P_s\left(\mathrm{Prox}_{-\alpha_k u_s}(\boldsymbol{v}_s(k))\right) & (i = s \in \mathcal{S}) \\ P_l\left(\boldsymbol{v}_l(k)\right) & (i = l \in \mathcal{L}) \end{cases}$$

5: **end for**
6: $k \leftarrow k + 1$
7: **end loop**

Assumption III.1 ensures that user i receives $\boldsymbol{x}_j(k)$ from its neighboring users $j \in \mathcal{N}_i(k)$. Moreover, Assumption III.3 means that user i has a set of weight parameters $w_{ij}(k)$ ($j \in \mathcal{V}$), which implies that user i can compute the weighted average $\boldsymbol{v}_i(k)$ in step 4 of Algorithm 1. When user i is source s, source s tries to maximize its utility function u_s over its constraint set C_s defined by (2). The simplest way to achieve this objective is for source s to implement a projected proximal algorithm [3], [25] using both the metric projection P_s onto C_s and the proximity operator $\mathrm{Prox}_{-\alpha u_s}$ ($\alpha > 0$). Assumptions II.1(ii) and II.2(i) guarantee that source s can use the computable metric projection P_s and the computable proximity operator $\mathrm{Prox}_{-\alpha u_s}$ ($\alpha > 0$). Since source s has $\boldsymbol{v}_s(k)$, source s can implement the projected proximal algorithm defined by $\boldsymbol{x}_s(k+1) = P_s(\mathrm{Prox}_{-\alpha_k u_s}(\boldsymbol{v}_s(k)))$. When user i is link l, link l tries to satisfy only the condition in its constraint set C_l defined by (1) since link l does not have utility of its own, i.e., $u_l(\boldsymbol{x}) = 0$. Accordingly, it is sufficient for link l to implement a projection algorithm using the metric projection P_l onto C_l. Assumption II.1(i) ensures that link l can use the computable metric projection P_l, and hence, link l can implement the projection algorithm defined by $\boldsymbol{x}_l(k+1) = P_l(\boldsymbol{v}_l(k))$. Since Assumption II.2(ii)

implies that $\mathrm{Prox}_{-\alpha_k u_l}(\boldsymbol{v}_l(k)) = \boldsymbol{v}_l(k)$, we can see that $\boldsymbol{x}_l(k+1) = P_l(\boldsymbol{v}_l(k)) = P_l(\mathrm{Prox}_{-\alpha_k u_l}(\boldsymbol{v}_l(k)))$.

The following is a convergence analysis of Algorithm 1. The proof of Theorem III.1 is given in Appendix A.

Theorem III.1 *Suppose that Assumptions II.1, II.2, III.1, III.2, and III.3 hold. Then the sequences $(\boldsymbol{v}_i(k))_{k\in\mathbb{N}}$ and $(\boldsymbol{x}_i(k))_{k\in\mathbb{N}}$ $(i \in \mathcal{V})$ generated by Algorithm 1 satisfy the following properties:*

(i) *If $\alpha_k := \alpha$ is chosen for all $k \in \mathbb{N}$, then there exist positive real numbers M_i $(i = 1, 2)$ such that*

$$\liminf_{k\to+\infty} \sum_{i\in\mathcal{V}} \mathrm{d}(\boldsymbol{x}_i(k), C)^2 \le M_1\alpha^2 \text{ and}$$
$$\limsup_{k\to+\infty} \sum_{i\in\mathcal{V}} u_i\left(P_C(\boldsymbol{v}_i(k))\right) \ge U^\star - M_2\alpha,$$

where $\mathrm{d}(\boldsymbol{x}, C) := \inf_{\boldsymbol{y}\in C} \|\boldsymbol{x} - \boldsymbol{y}\|$ $(\boldsymbol{x} \in \mathbb{R}^S)$ and $U^\star$ is the optimal value of Problem II.1.

(ii) *If $(\alpha_k)_{k\in\mathbb{N}}$ is such that $\sum_{k=0}^{+\infty} \alpha_k = +\infty$ and $\sum_{k=0}^{+\infty} \alpha_k^2 < +\infty$, then $(\boldsymbol{x}_i(k))_{k\in\mathbb{N}}$ $(i \in \mathcal{V})$ converges to a solution $\boldsymbol{x}^\star$ of Problem II.1, as well as satisfying that*

$$\sum_{i\in\mathcal{V}} \mathrm{d}\left(\frac{1}{k}\sum_{t=1}^{k} \boldsymbol{x}_i(t), C\right)^2 = \mathcal{O}\left(\frac{1}{k}\right). \tag{7}$$

Assume that (A) *there exists a positive real number c such that $c\sum_{i\in\mathcal{V}} \|\boldsymbol{v}_i(k) - \boldsymbol{x}^\star\|^2 \le \sum_{i\in\mathcal{V}} \mathrm{d}(\boldsymbol{v}_i(k), C)^2$ for all $k \in \mathbb{N}$. Then the following hold:*

(a) *If $\alpha_k = c_1/k$ for some positive real number c_1, then*

$$\sum_{i\in\mathcal{V}} u_i\left(\frac{1}{k}\sum_{t=1}^{k} P_C(\boldsymbol{v}_i(t))\right) \ge U^\star - \mathcal{O}\left(\frac{1+\log k}{k}\right);$$

(b) *If $\alpha_k = c_2/(k+1)$ for some positive real number c_2, then*

$$\sum_{i\in\mathcal{V}} u_i\left(\frac{2}{k(k+1)}\sum_{t=1}^{k} tP_C(\boldsymbol{v}_i(t))\right) \ge U^\star - \mathcal{O}\left(\frac{1}{k}\right).$$

Theorem III.1(i) implies that $\liminf_{k\to+\infty} \mathrm{d}(\boldsymbol{x}_i(k), C)^2 \le M_1\alpha^2$ $(i \in \mathcal{V})$. Accordingly, we can expect that the sequence $(\boldsymbol{x}_i(k))_{k\in\mathbb{N}}$ $(i \in \mathcal{V})$ generated by Algorithm 1 with a small step size α approximates a point in the constraint set of the NUM problem. Moreover, Theorem III.1(i) implies that, when α is small, $\boldsymbol{v}_i(k)$ $(i \in \mathcal{V})$ may approximate a solution to the NUM problem.

Meanwhile, Theorem III.1(ii) guarantees that, if we can choose a diminishing step size $(\alpha_k)_{k\in\mathbb{N}}$ (e.g., $\alpha_k := 1/k$ $(k \in \mathbb{N}\backslash\{0\})$), then the sequence $(\boldsymbol{x}_i(k))_{k\in\mathbb{N}}$ of each user i converges to a solution of Problem II.1, i.e., each source s and each link l can find the optimal resource allocation in a distributed manner. From (7), we have that

$$\mathrm{d}\left(\frac{1}{k}\sum_{t=1}^{k} \boldsymbol{x}_i(t), C\right)^2 = \mathcal{O}\left(\frac{1}{k}\right) \quad (i \in \mathcal{V}),$$

which implies that the squared distance between the average of $(\boldsymbol{x}_i(t))_{t=1}^k$ and C converges at an $\mathcal{O}(1/k)$ rate. In general, we have that $\mathrm{d}(\boldsymbol{v}_i(k), C) := \inf_{\boldsymbol{x}\in C} \|\boldsymbol{v}_i(k) - \boldsymbol{x}\| \le \|\boldsymbol{v}_i(k) - \boldsymbol{x}^\star\|$ $(i \in \mathcal{V}, k \in \mathbb{N})$, i.e., $\sum_{i\in\mathcal{V}} \mathrm{d}(\boldsymbol{v}_i(k), C)^2 = \mathcal{O}(\sum_{i\in\mathcal{V}} \|\boldsymbol{v}_i(k) - \boldsymbol{x}^\star\|^2)$. Meanwhile, condition (A) implies that $c\sum_{i\in\mathcal{V}} \|\boldsymbol{v}_i(k) - \boldsymbol{x}^\star\|^2 \le \sum_{i\in\mathcal{V}} \mathrm{d}(\boldsymbol{v}_i(k), C)^2 \le \sum_{i\in\mathcal{V}} \|\boldsymbol{v}_i(k) - \boldsymbol{x}^\star\|^2$ $(k \in \mathbb{N})$, i.e., $\sum_{i\in\mathcal{V}} \mathrm{d}(\boldsymbol{v}_i(k), C)^2 = \Theta(\sum_{i\in\mathcal{V}} \|\boldsymbol{v}_i(k) - \boldsymbol{x}^\star\|^2)$ [15, p.19]. Theorem III.1(ii) (a) and (b) indicate that, under the assumption that $\sum_{i\in\mathcal{V}} \mathrm{d}(\boldsymbol{v}_i(k), C)^2$ is bounded not only above but also below by $\sum_{i\in\mathcal{V}} \|\boldsymbol{v}_i(k) - \boldsymbol{x}^\star\|^2$ asymptotically[4], the optimality error between the sum of the values of u_i $(i \in \mathcal{V})$ at the average of $(P_C(\boldsymbol{v}_i(t)))_{t=1}^k$ and $U^\star$ converges at a rate of $\mathcal{O}((1+\log k)/k)$ or $\mathcal{O}(1/k)$. The DNFC algorithm [30, Algorithm 2] achieves an $\mathcal{O}(1/k)$ convergence rate (see (14)).

C. Projected Subgradient Algorithm

The following is a distributed subgradient algorithm for solving Problem II.1 (see [16, (2a), (2b)] for a distributed random projection algorithm for smooth convex optimization). Algorithm 2 is obtained by replacing $\mathrm{Prox}_{-\alpha_k u_i}(\boldsymbol{v}_i(k))$ in Algorithm 1 with $\boldsymbol{v}_i(k) - \alpha_k \boldsymbol{g}_i(k)$, where $\boldsymbol{g}_i(k)$ stands for the subgradient of $-u_i$ at $\boldsymbol{v}_i(k)$ and $\boldsymbol{g}_l(k) = \boldsymbol{0}$ $(l \in \mathcal{L}, k \in \mathbb{N})$ (by Assumption II.2(ii)).

Algorithm 2 Projected subgradient algorithm

Require: $(\alpha_k)_{k\in\mathbb{N}} \subset (0, +\infty)$

$k \leftarrow 0$, $\boldsymbol{x}_i(0) \in \mathbb{R}^S$ $(i \in \mathcal{V} := \mathcal{S} \cup \mathcal{L})$

loop

  for $i = 1$ to $i = V := S + L$ **do**

$$\boldsymbol{v}_i(k) := \sum_{j\in\mathcal{N}_i(k)} w_{ij}(k)\boldsymbol{x}_j(k)$$
$$\boldsymbol{g}_i(k) \in \partial(-u_i)(\boldsymbol{v}_i(k))$$
$$x_i(k+1) := P_i\left(\boldsymbol{v}_i(k) - \alpha_k \boldsymbol{g}_i(k)\right) = \begin{cases} P_s\left(\boldsymbol{v}_s(k) - \alpha_k \boldsymbol{g}_s(k)\right) & (i = s \in \mathcal{S}) \\ P_l\left(\boldsymbol{v}_l(k)\right) & (i = l \in \mathcal{L}) \end{cases}$$

  end for

  $k \leftarrow k + 1$

end loop

The following convergence analysis of Algorithm 2 is the same as that in Theorem III.1. We can prove Theorem III.2 by referring to the proof of Theorem III.1 and the results in [11, Section 4] (see Appendix A).

Theorem III.2 *Suppose that the assumptions in Theorem III.1 hold. Then, the sequences $(\boldsymbol{v}_i(k))_{k\in\mathbb{N}}$ and $(\boldsymbol{x}_i(k))_{k\in\mathbb{N}}$ $(i \in \mathcal{V})$ generated by Algorithm 2 satisfy the properties in Theorem III.1(i) and (ii).*

D. Existing Algorithms for Nonsmooth Convex Optimization

This subsection surveys optimization algorithms for solving Problem II.1 and related work. Parikh and Boyd proposed the following algorithm [25, Subsection 5.4, (5.14)] that can be

[4]We verified that $(\sum_{i\in\mathcal{V}} \|\boldsymbol{v}_i(k) - \boldsymbol{x}^\star\|^2)_{k\in\mathbb{N}}$ generated by Algorithms 1 and 2 used in Section IV has almost the same convergence rate as $(\sum_{i\in\mathcal{V}} \mathrm{d}(\boldsymbol{v}_i(k), C)^2)_{k\in\mathbb{N}}$ and Algorithms 1 and 2 satisfy (A) for all $k \le 10^3$ with $c = 10^{-3}$.

applied to Problem II.1: Suppose that user i has $\boldsymbol{x}_i(k)$, $\boldsymbol{v}_i(k)$, $\boldsymbol{x}(k+1)$, $\boldsymbol{z}(k)$, and $\boldsymbol{v}(k)$ at time k. User i computes

$$\begin{aligned}
&\boldsymbol{x}_i(k+1) := \mathrm{Prox}_{-\lambda u_i}\left(\boldsymbol{x}_i(k) - \boldsymbol{z}(k) - \boldsymbol{v}(k)\right),\\
&\boldsymbol{z}(k+1) := P_C\left(\boldsymbol{x}(k+1) + \boldsymbol{v}(k)\right),\\
&\boldsymbol{v}_i(k+1) := \boldsymbol{v}_i(k) + \boldsymbol{x}_i(k+1) - \boldsymbol{z}(k+1),
\end{aligned} \tag{8}$$

where $\lambda > 0$. Bertsekas proposed three incremental proximal algorithms [3, (19), (20), (21)] for minimizing $\sum_{i\in\mathcal{V}}(f_i(\boldsymbol{x}) + h_i(\boldsymbol{x}))$ subject to $\boldsymbol{x} \in C$, where $f_i\colon \mathbb{R}^S \to \mathbb{R}$ and $h_i\colon \mathbb{R}^S \to \mathbb{R}$ $(i \in \mathcal{V})$ are convex, and the proximal operator of f_i and the subdifferential of h_i can be computed efficiently. For example, algorithm (20) in [3] is as follows:

$$\begin{aligned}
&\boldsymbol{z}(k) := \mathrm{Prox}_{\lambda_k f_i}\left(\boldsymbol{x}(k)\right),\\
&\boldsymbol{x}(k+1) := P_C\left(\boldsymbol{z}(k) - \lambda_k \tilde{\nabla} h_i(\boldsymbol{z}(k))\right),
\end{aligned} \tag{9}$$

where $(\lambda_k)_{k\in\mathbb{N}} \subset (0,1)$ satisfies that $\lim_{k\to+\infty}\lambda_k = 0$ and $\sum_{k=0}^{+\infty}\lambda_k = +\infty$, $\{f_i, h_i\}$ are chosen for iteration in a cyclic order or a randomized order [3, p.170, (1), (2)], and $\tilde{\nabla} h_i(\boldsymbol{x}) \in \partial h_i(\boldsymbol{x})$ $(i \in \mathcal{V}, \boldsymbol{x} \in \mathbb{R}^S)$. Under certain assumptions, the sequence $(\boldsymbol{x}(k))_{k\in\mathbb{N}}$ generated by algorithm (9) converges to some minimizer of $\sum_{i\in\mathcal{V}}(f_i + h_i)$ over C [3, Propositions 6 and 9]. When algorithms (8) and (9) are applied to Problem II.1, it is necessary to compute the metric projection P_C onto C defined by (3). Since the form of (3) is not always simple, it would be difficult to apply algorithms using P_C at each iteration to Problem II.1.

Nedić and Ozdaglar [23], [24] studied the dual problem of Problem II.1 as follows:

$$\begin{aligned}
&\text{maximize } q(\boldsymbol{\mu}) := \sum_{s\in\mathcal{S}} \max_{x_s\in I_s}\{u_s(x_s) - x_s\mu_s\} + \sum_{l\in\mathcal{L}}\mu_l c_l\\
&\text{subject to } \boldsymbol{\mu} := (\mu_i)_{i\in\mathcal{V}} \in \mathbb{R}_+^V.
\end{aligned} \tag{10}$$

They proposed the following dual subgradient and primal-dual subgradient methods [23, Subsection 2.2, (15), (16)] to solve the NUM problem (see also [24, (16), (17)]): given $\boldsymbol{\mu}(k) := (\mu_i(k))_{i\in\mathcal{V}} \in \mathbb{R}_+^V$,

$$\begin{aligned}
&x_s(k) \in \operatorname*{argmax}_{x_s\in I_s}\{u_s(x_s) - x_s\mu_s(k)\} + \sum_{l\in\mathcal{L}}\mu_l(k)c_l,\\
&\boldsymbol{\mu}(k+1) := \left[\boldsymbol{\mu}(k) + \gamma_k \boldsymbol{g}(\boldsymbol{x}(k))\right]^+,\\
&\tilde{\boldsymbol{x}}(k) := \frac{\sum_{s\in\mathcal{S}}\alpha_s \boldsymbol{x}(k)}{\sum_{s\in\mathcal{S}}\alpha_s},
\end{aligned} \tag{11}$$

where $g_l(\boldsymbol{x}) := \sum_{s\in\mathcal{S}(l)} x_s - c_l$ $(l \in \mathcal{L}, \boldsymbol{x} := (x_s)_{s\in\mathcal{S}} \in \mathbb{R}^S)$, $g_s(\boldsymbol{x}) := 0$ $(s \in \mathcal{S}, \boldsymbol{x} \in \mathbb{R}^S)$, $\boldsymbol{g}(\boldsymbol{x}) := (g_i(\boldsymbol{x}))_{i\in\mathcal{V}}^\top$ $(\boldsymbol{x} \in \mathbb{R}^S)$, $(\gamma_k)_{k\in\mathbb{N}}, (\alpha_s)_{s\in\mathcal{S}} \subset (0, +\infty)$, and $\boldsymbol{\mu}^+ := (\max\{0, \mu_i\})_{i\in\mathcal{V}}^\top$ $(\boldsymbol{\mu} := (\mu_i)_{i\in\mathcal{V}} \in \mathbb{R}^V)$. Proposition 2 in [23] provides the lower and upper bounds on the primal cost $\sum_{s\in\mathcal{S}} u_s$ of $\tilde{\boldsymbol{x}}(k)$ defined by (11) under certain assumptions. The update of $\boldsymbol{\mu}(k)$ defined by algorithm (11) uses all the values of the c_ls since algorithm (11) uses $g_l(\boldsymbol{x}) := \sum_{s\in\mathcal{S}(l)} x_s - c_l$ $(l \in \mathcal{L}, \boldsymbol{x} := (x_s)_{s\in\mathcal{S}} \in \mathbb{R}^S)$. Hence, algorithm (11) can be implemented by "network providers" [23, p.1761] knowing all of the values of the c_ls. However, the existence of such "network providers" might not be realistic in modern communication networks. Meanwhile, each link l in Algorithms 1 and 2 uses only the information of its own link (i.e., the closed-form expression of C_l) and each source s uses only its own utility function and constraint set (i.e., the closed-form expressions of u_s and C_s). As a result, they can find the optimal resource allocation using Algorithms 1 and 2 (Theorems III.1 and III.2).

Yu and Neely proposed a decentralized dual subgradient algorithm [30, Algorithm 2] that can be applied to the multipath NUM problem including Problem II.1. The following is the DNFC algorithm [30, Algorithm 2] for Problem II.1. Given a point $Q_l(k)$, each link l receives $x_s(k)$ $(s \in \mathcal{S}(l))$ and updates $Q_l(k+1)$ and $Y_l(k+1)$ defined as follows:

$$\begin{aligned}
&Q_l(k+1)\\
&\quad := \max\left\{-\sum_{s\in\mathcal{S}(l)} x_s(k) + c_l, Q_l(k) + \sum_{s\in\mathcal{S}(l)} x_s(k) - c_l\right\},\\
&Y_l(k+1) := Q_l(k+1) + \sum_{s\in\mathcal{S}(l)} x_s(k) - c_l.
\end{aligned}$$

Link l transmits $Y_l(k+1)$ to source s $(s \in \mathcal{S}(l))$. Meanwhile, given points $x_s(k-1)$, $y_s(k-1)$, $R_s(k)$, and $Z_s(k)$, each source s receives $Y_l(k)$ $(l \in \mathcal{L}(s))$ and updates $x_s(k)$ defined as follows:

$$x_s(k) := \left[x_s(k-1) - \frac{1}{2\alpha}\left(\sum_{l\in\mathcal{L}(s)} Y_l(k) - Z_s(k)\right)\right]_0^{M_s},$$

where $\alpha > 0$ and $[z]_a^b := \min\{\max\{z, a\}, b\}$ $(z, a, b \in \mathbb{R})$. Source s transmits $x_s(k)$ to link l $(l \in \mathcal{L}(s))$ and updates $y_s(k)$ as follows:

$$y_s(k) := \operatorname*{argmin}_{y_s\in I_s}\left\{-u_s(y_s) + Z_s(k)y_s + \alpha(y_s - y_s(k-1))^2\right\}. \tag{12}$$

Source s updates $R_s(k+1)$ and $Z_s(k+1)$ by

$$\begin{aligned}
&R_s(k+1) := \max\left\{-y_s(k) + x_s(k), R_s(k) + y_s(k) - x_s(k)\right\},\\
&Z_s(k+1) := R_s(k+1) + y_s(k) - x_s(k).
\end{aligned}$$

When $\alpha \geq (2S + \sum_{s\in\mathcal{S}} d_s)/2$ is chosen, where d_s $(s \in \mathcal{S})$ is the length of path of source s, the sequence $(\bar{\boldsymbol{y}}(k))_{k\in\mathbb{N}} := ((\bar{y}_s(k))_{s\in\mathcal{S}})_{k\in\mathbb{N}}$ generated by

$$\bar{y}_s(k) := \frac{1}{k}\sum_{t=0}^{k-1} y_s(t) \qquad (s \in \mathcal{S}), \tag{13}$$

where $y_s(k)$ $(s \in \mathcal{S}, k \in \mathbb{N})$ is defined as in (12), converges to a solution of Problem II.1 at the following rate [30, Theorem 3, (23)]:

$$\sum_{s\in\mathcal{S}} u_s\left(\frac{1}{k}\sum_{t=0}^{k-1} y_s(t)\right) \geq U^\star - \mathcal{O}\left(\frac{1}{k}\right) \tag{14}$$

(see Theorem III.1(ii) for the rate of convergence of Algorithms 1 and 2). The DNFC algorithm defined by (13) can work under the assumptions considered in the present paper. We can see that Algorithms 1 and 2 use the average $\boldsymbol{v}_i(k)$ of $\boldsymbol{x}_j(k)$ $(j \in \mathcal{N}_i(k), k \geq 0)$ with the weight parameters $w_{ij}(k)$ defined by Assumption III.3, whereas the DNFC algorithm uses the average $\bar{y}_s(k)$ of $(y_s(t))_{t=0}^{k-1}$ defined by (12).

IV. Numerical Experiments

We consider the network model in Figure 1, which is based on [9, Figure 3.12]. As shown, the network comprises two dense networks that are connected by a bridge, and the number of nodes in each of the two networks is 20. The number of sources in each of the two networks is 2 and each source uses 3 links chosen randomly. The number of sources using the bridge and the two networks is 6 and each source uses 7 links chosen randomly, where the graph with sources and links in the network is modified to satisfy Assumption III.2. The capacity of each link is randomly chosen from $(0, 1)$. The maximum allowed rate for each source is randomly chosen from $(0, 1)$.

The experiments were conducted on a Mac Pro (Late 2013) computer whose processor is a 3 GHz 8-Core Intel Xeon E5 CPU and whose main memory is 32 GB 1866 MHz DDR3 RAM. The algorithms used in the experiments were coded in Python 3 with the NumPy 1.14.3 and SciPy 1.1.0 packages.

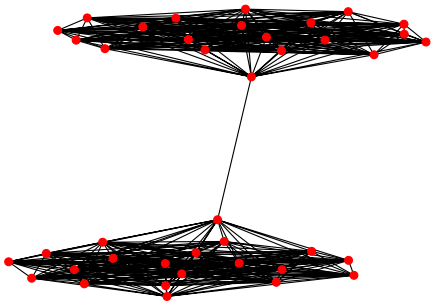


Fig. 1: Network model based on [9, Figure 3.12]

Source parameter t_s for s was randomly chosen from $\{0, 1\}$. If $t_s = 0$, then the utility function of source s is defined by (5) with $w_s \in [2, 5)$ and $a_s \in [5, 20)$ randomly chosen, where $b_s := -w_s a_s$. If $t_s = 1$, then the utility function of source s is defined by (4) with $p_s := 1$ and a randomly chosen $w_s \in [15, 20)$. Obviously, Assumption II.2(i) is satisfied. We define $u_l(\boldsymbol{x}) := 0$ $(l \in \mathcal{L}, \boldsymbol{x} \in \mathbb{R}^S)$ to satisfy Assumption II.2(ii). We assume that user i $(i \in \mathcal{V} := \mathcal{S} \cup \mathcal{L})$ satisfies the standard assumption II.1, i.e., user i knows only its own private information.

We verify how Algorithms 1 and 2 are affected by step sizes $(\alpha_k)_{k\in\mathbb{N}}$. In the experiments, we used

$$\alpha_k := 1, 10^{-1}, 10^{-2},$$
$$\alpha_k := \frac{1}{k+1}, \frac{10^{-1}}{k+1}, \frac{10^{-2}}{k+1},$$

which satisfy the conditions in Theorems III.1 and III.2. The weight parameters $w_{ij}(k)$ $(i \in \mathcal{V}, j \in \mathcal{N}_i(k))$[5] were set to satisfy Assumptions III.1, III.2, and III.3. Accordingly, Theorems III.1 and III.2 guarantee that, when both Algorithms 1 and 2 with diminishing step sizes are implemented, each user in the network can find the optimal resource allocation. Moreover, we compare the behaviors of Algorithms 1 and 2 with that of the DNFC algorithm (13) [30, Algorithm 2] with parameters satisfying the conditions in [30, Theorem 3].

Two performance measures were used in the experiments: for $\boldsymbol{x}_i(0) := \boldsymbol{0}$ $(i \in \mathcal{V})$ and all $k \in \mathbb{N}$,

$$D(k) := \begin{cases} \sum_{i\in\mathcal{V}} \|\bar{\boldsymbol{x}}(k) - P_i(\bar{\boldsymbol{x}}(k))\| & \text{(Algorithms 1 and 2)} \\ \sum_{i\in\mathcal{V}} \|\bar{\boldsymbol{y}}(k) - P_i(\bar{\boldsymbol{y}}(k))\| & \text{(DNFC),} \end{cases}$$

$$U(k) := \begin{cases} \sum_{s\in\mathcal{S}} u_s(x_{s,s}(k)) & \text{(Algorithms 1 and 2)} \\ \sum_{s\in\mathcal{S}} u_s(\bar{y}_s(k)) & \text{(DNFC),} \end{cases}$$

where $\boldsymbol{x}_s(k) := (x_{s,t}(k))_{t\in\mathcal{S}} \in \mathbb{R}^S$ $(s \in \mathcal{S}, k \in \mathbb{N})$ is the point generated by one of Algorithms 1 and 2 and $\bar{\boldsymbol{x}}(k) := (x_{s,s}(k))_{s\in\mathcal{S}} \in \mathbb{R}^S$. If $(D(k))_{k\in\mathbb{N}}$ converges to 0, then the algorithms converge to a point in the constraint set $\bigcap_{i\in\mathcal{V}} \mathrm{Fix}(P_i) = \bigcap_{i\in\mathcal{V}} C_i = C$. $U(k)$ $(k \in \mathbb{N})$ is used to evaluate the objective function in Problem II.1. The elapsed time for the algorithms used in the experiments was obtained by using the `timeit` module in Python. Values $k = 10^2, 10^3$, and 10^4 were used in the stopping condition for each of the algorithms. We verified that the optimal value of Problem II.1 is $U^\star \approx 19.61432375$ by using Sequential Least SQuares Programming (SLSQP) [14], which is a centralized optimization solver in the SciPy 1.1.0 package.

Table I lists the elapsed time and values of $D(k)$ and $U(k)$ $(k = 10^2, 10^3)$ for the DNFC algorithm (13) [30, Algorithm 2] and Algorithms 1 and 2. It is necessary to verify whether the algorithms converge quickly to a point in C (i.e., $D(k) \approx 0$) because the set C is the constraint set in the NUM problem. First, we compare the values of $D(k)$ $(k = 10^2, 10^3)$ for Algorithm 1 with the ones for Algorithm 2. Table I shows that, when the step size was fixed, Algorithms 1 and 2 had almost the same behaviors. Algorithms 1 and 2 with diminishing step sizes satisfied $D(10^3) \approx 0$, whereas Algorithms 1 and 2 with constant step sizes satisfied $D(k) > 1.7$ $(k = 10^2, 10^3)$. The elapsed time of Algorithm 1 was almost the same as that of Algorithm 2. From the above discussion, Algorithms 1 and 2 with diminishing step sizes should be used to solve Problem II.1. This is supported by the convergence analyses of Algorithms 1 and 2 (Theorems III.1 and III.2) guaranteeing that Algorithms 1 and 2 using a diminishing step size converge to a solution of Problem II.1. Meanwhile, Table I shows that, although the DNFC algorithm required much time, $(D(k))_{k=0}^{1000}$ generated by the DNFC algorithm converged to 0. Hence, we can see that the DNFC algorithm performed better than Algorithms 1 and 2 with constant step sizes.

Next, we compare the values of $U(k)$ $(k = 10^2, 10^3)$ for the algorithms in the experiments. Table I indicates that the DNFC algorithm converged to a solution of Problem II.1, as promised in [30, Theorem 3]. Similarly, Algorithms 1 and 2 with $\alpha_k = 1/(k+1), 10^{-1}/(k+1)$ converged to a solution of Problem II.1, as promised in Theorems III.1 and III.2. In addition, $U(10^3)$ (≈ 4) generated by either of Algorithms 1 and 2 with $\alpha_k = 10^{-2}/(k+1)$ was less than $U^\star$, whereas they converged to a point in C faster than other algorithms. Algorithms 1 and 2 with constant step sizes had $U(10^3) > 30$

[5] See https://github.com/iiduka-researches/misc/blob/master/misc/matrix.py for the method of setting $w_{ij}(k)$ $(i \in \mathcal{V}, j \in \mathcal{N}_i(k))$.

TABLE I: Elapsed time and values of $D(k)$ and $U(k)$ $(k = 10^2, 10^3)$ for the DNFC algorithm, and Algorithms 1 and 2

(a) $k = 10^2$

Algorithm	Time [s]	$D(10^2)$	$U(10^2)$
DNFC [30]	0.91254	32.38765	95.30908
Alg. 1 ($\alpha_k = 1$)	0.16692	6.70352	42.78859
Alg. 2 ($\alpha_k = 1$)	0.16309	6.70352	42.78859
Alg. 1 ($\alpha_k = 10^{-1}$)	0.17012	6.37677	42.43657
Alg. 2 ($\alpha_k = 10^{-1}$)	0.16600	6.37677	42.43657
Alg. 1 ($\alpha_k = 10^{-2}$)	0.16092	1.89796	33.34397
Alg. 2 ($\alpha_k = 10^{-2}$)	0.15328	1.93210	33.56496
Alg. 1 ($\alpha_k = 1/(k+1)$)	0.16306	2.10725	33.79944
Alg. 2 ($\alpha_k = 1/(k+1)$)	0.15987	2.14690	34.03348
Alg. 1 ($\alpha_k = 10^{-1}/(k+1)$)	0.15510	0.17539	18.21854
Alg. 2 ($\alpha_k = 10^{-1}/(k+1)$)	0.14941	0.18222	18.46435
Alg. 1 ($\alpha_k = 10^{-2}/(k+1)$)	0.14947	0.00793	4.93398
Alg. 2 ($\alpha_k = 10^{-2}/(k+1)$)	0.14553	0.00796	5.01410

(b) $k = 10^3$

Algorithm	Time [s]	$D(10^3)$	$U(10^3)$
DNFC [30]	10.66114	0.49226	17.88063
Alg. 1 ($\alpha_k = 1$)	1.63952	6.70352	42.78859
Alg. 2 ($\alpha_k = 1$)	1.60649	6.70352	42.78859
Alg. 1 ($\alpha_k = 10^{-1}$)	1.71241	6.37677	42.43657
Alg. 2 ($\alpha_k = 10^{-1}$)	1.66357	6.37677	42.43657
Alg. 1 ($\alpha_k = 10^{-2}$)	1.60314	1.71861	32.98997
Alg. 2 ($\alpha_k = 10^{-2}$)	1.55142	1.75692	33.21719
Alg. 1 ($\alpha_k = 1/(k+1)$)	1.48479	0.34058	23.87111
Alg. 2 ($\alpha_k = 1/(k+1)$)	1.45622	0.34550	23.91450
Alg. 1 ($\alpha_k = 10^{-1}/(k+1)$)	1.47941	0.02834	13.84689
Alg. 2 ($\alpha_k = 10^{-1}/(k+1)$)	1.44119	0.02800	13.93110
Alg. 1 ($\alpha_k = 10^{-2}/(k+1)$)	1.41824	0.00069	4.18878
Alg. 2 ($\alpha_k = 10^{-2}/(k+1)$)	1.40255	0.00069	4.23956

and $D(10^3) > 1.7$; i.e., they did not approximate a solution to Problem II.1 until the stopping conditions were satisfied.

We next verify the behaviors of the sequences $(D(k))_{k=0}^{1000}$ and $(U(k))_{k=0}^{1000}$ generated by the algorithms in the experiments. Figures 2–4 show the behaviors of the DNFC algorithm and Algorithms 1 and 2 with constant step sizes. The sequence $(D(k))_{k=50}^{1000}$ generated by the DNFC algorithm decreased and the DNFC algorithm converged to a solution of Problem II.1, as seen in Table I. Meanwhile, these figures indicate that Algorithms 1 and 2 with constant step sizes did not converge to a point in C (see also Table I).

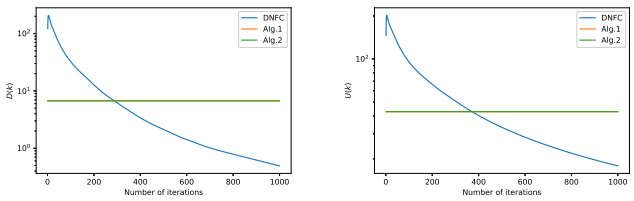
Figures 5–7 show the behaviors of the DNFC algorithm and Algorithms 1 and 2 with diminishing step sizes. The figures indicate that $(D(k))_{k=0}^{1000}$ generated by the DNFC algorithm and Algorithms 1 and 2 converged to 0. In particular, Algorithms 1 and 2 converged to a point in C faster than the DNFC algorithm. Moreover, Figure 5 and Table I show that Algorithms 1 and 2 with $\alpha_k = 1/(k+1)$ performed better than the DNFC algorithm. Algorithms 1 and 2 with $\alpha_k = 10^{-1}/(k+1), 10^{-2}/(k+1)$ converged to a point in C faster than Algorithms 1 and 2 with $\alpha_k = 1/(k+1)$. However, the value of $U(10^3)$ generated by either of Algorithms 1 and 2 was less with $\alpha_k = 10^{-1}/(k+1), 10^{-2}/(k+1)$ than with $\alpha_k = 1/(k+1)$ (Table I and Figures 5–7). Accordingly, we can see that Algorithms 1 and 2 performed better with $\alpha_k = 1/(k+1)$ than with $\alpha_k = 10^{-1}/(k+1), 10^{-2}/(k+1)$.

We also verify whether Algorithms 1 and 2 with $\alpha_k = 10^{-1}/(k+1), 10^{-2}/(k+1)$ approximate a solution to Problem II.1 for a large enough iteration k. Tables I and II show that Algorithms 1 and 2 with $\alpha_k = 10^{-1}/(k+1), 10^{-2}/(k+1)$ converged to a point in C and that they increased $U(k)$ little by little. Hence, we can see that, although Algorithms 1 and 2 required more time with $\alpha_k = 10^{-1}/(k+1), 10^{-2}/(k+1)$ than with $\alpha_k = 1/(k+1)$, Algorithms 1 and 2 with $\alpha_k = 10^{-1}/(k+1), 10^{-2}/(k+1)$ converged to a solution of Problem II.1.

TABLE II: Elapsed time and values of $D(10^4)$ and $U(10^4)$ for Algorithms 1 and 2 with $\alpha_k = 10^{-1}/(k+1), 10^{-2}/(k+1)$

Algorithm	Time [s]	$D(10^4)$	$U(10^4)$
Alg. 1 ($\alpha_k = 10^{-1}/(k+1)$)	14.19869	0.00247	15.17346
Alg. 2 ($\alpha_k = 10^{-1}/(k+1)$)	13.73651	0.00260	15.25161
Alg. 1 ($\alpha_k = 10^{-2}/(k+1)$)	13.75821	0.00007	4.79908
Alg. 2 ($\alpha_k = 10^{-2}/(k+1)$)	13.26862	0.00007	4.84906

From the above discussion, we can conclude that the performances of Algorithms 1 and 2 depend on step size. The comparisons of the DNFC algorithm with Algorithms 1 and 2 showed that the DNFC algorithm performs better than Algorithms 1 and 2 with constant step sizes, whereas Algorithms 1 and 2 with diminishing step sizes perform better than the DNFC algorithm. In particular, Algorithms 1 and 2 with small diminishing step sizes (e.g., $\alpha_k = 10^{-1}/(k+1), 10^{-2}/(k+1)$) converge to a point in the absolute constraint set quickly, and Algorithms 1 and 2 with large diminishing step sizes (e.g., $\alpha_k = 1/(k+1)$) approximate a solution to Problem II.1 quickly.

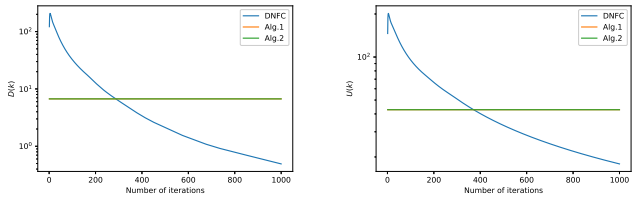


(a) Evaluation of $D(k)$ (b) Evaluation of $U(k)$

Fig. 2: Behaviors of $D(k)$ and $U(k)$ for the DNFC algorithm, and Algorithms 1 and 2 with $\alpha_k := 1$ (Algorithm 1 had almost the same behavior as Algorithm 2)

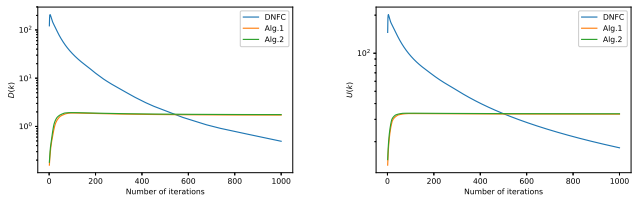
V. Conclusion

This paper proposed two distributed optimization algorithms, a projected proximal algorithm and a projected subgradient algorithm, for solving the NUM problem under the constraint that each source has its own nonsmooth concave utility function and each link has its own capacity constraint. The algorithms are each guaranteed to converge to the solution



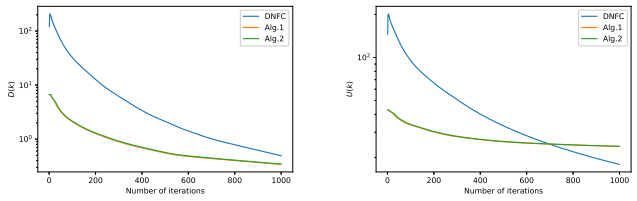
(a) Evaluation of $D(k)$ (b) Evaluation of $U(k)$

Fig. 3: Behaviors of $D(k)$ and $U(k)$ for the DNFC algorithm, and Algorithms 1 and 2 with $\alpha_k := 10^{-1}$ (Algorithm 1 had almost the same behavior as Algorithm 2)



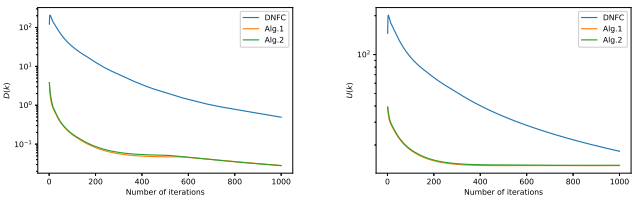
(a) Evaluation of $D(k)$ (b) Evaluation of $U(k)$

Fig. 4: Behaviors of $D(k)$ and $U(k)$ for the DNFC algorithm, and Algorithms 1 and 2 with $\alpha_k := 10^{-2}$ (Algorithm 1 had almost the same behavior as Algorithm 2)



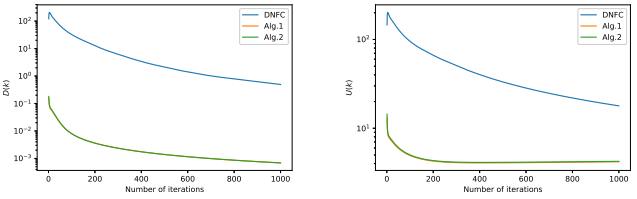
(a) Evaluation of $D(k)$ (b) Evaluation of $U(k)$

Fig. 5: Behaviors of $D(k)$ and $U(k)$ for the DNFC algorithm, and Algorithms 1 and 2 with $\alpha_k := 1/(k+1)$ (Algorithm 1 had almost the same behavior as Algorithm 2)



(a) Evaluation of $D(k)$ (b) Evaluation of $U(k)$

Fig. 6: Behaviors of $D(k)$ and $U(k)$ for the DNFC algorithm, and Algorithms 1 and 2 with $\alpha_k := 10^{-1}/(k+1)$ (Algorithm 1 had almost the same behavior as Algorithm 2)



(a) Evaluation of $D(k)$ (b) Evaluation of $U(k)$

Fig. 7: Behaviors of $D(k)$ and $U(k)$ for the DNFC algorithm, and Algorithms 1 and 2 with $\alpha_k := 10^{-2}/(k+1)$ (Algorithm 1 had almost the same behavior as Algorithm 2)

of the NUM problem under certain assumptions. Numerical experiments were performed to demonstrate the performances of the two algorithms and the existing DNFC algorithm [30] for the NUM problem. The numerical results indicated that the algorithms were able to find the optimal resource allocation. In particular, the proposed algorithms with diminishing step sizes are well suited for finding the solution to the NUM problem.

Appendix A
Proofs of Theorems III.1 and III.2

Let us consider the following algorithm [11, Algorithm 3.1]. User i receives $\boldsymbol{x}_j(k)$ from its neighboring users $j \in \mathcal{N}_i(k)$ and computes $\boldsymbol{x}_i(k+1)$ defined as follows:

$$\begin{aligned} \boldsymbol{v}_i(k) &:= \sum_{j\in\mathcal{N}_i(k)} w_{ij}(k)\boldsymbol{x}_j(k), \\ \boldsymbol{x}_i(k+1) &:= \mathsf{P}_{i,\xi}\left(\mathrm{Prox}_{\alpha_k f_i}(\boldsymbol{v}_i(k))\right), \end{aligned} \tag{15}$$

where $f_i := -u_i$ $(i \in \mathcal{V})$, $\mathsf{P}_{i,\xi}$ $(i \in \mathcal{V})$ is the metric projection onto a constraint set $C_{i,\xi}$ randomly selected from $\{C_{i,j}\}_{j=1}^{J_i}$, and $C_i := \bigcap_{j=1}^{J_i} C_{i,j}$. Since Problem II.1 is considered under the special case that $C_i = C_{i,j}$ for all $i \in \mathcal{V}$ and all $j = 1, 2, \ldots, J_i$, we consider algorithm (15) when $\mathsf{P}_{i,j} := P_i$ $(i \in \mathcal{V}, j = 1, 2, \ldots, J_i)$, which is the same as in Algorithm 1. Accordingly, we can apply the results in [11] for algorithm (15) to Algorithm 1.

We first give a lemma.

Lemma A.1 *Suppose that Assumptions II.1, II.2, and III.3 hold and define* $\hat{D}(k) := \sum_{i\in\mathcal{V}} \mathrm{d}(\boldsymbol{x}_i(k), C)^2$, $\bar{D}(k) := \sum_{i\in\mathcal{V}} \|\boldsymbol{x}_i(k) - \boldsymbol{x}^\star\|^2$, $\boldsymbol{z}_i(k) := P_C(\boldsymbol{v}_i(k))$, *and* $\boldsymbol{z}(k) := \sum_{i\in\mathcal{V}} f_i(\boldsymbol{z}_i(k))$ *for all* $k \in \mathbb{N}$, *where* $\boldsymbol{x}^\star \in C$ *is a solution to Problem II.1. Then, there exist* $N_1 \in (0,1)$ *and* $N_i > 0$ $(i = 2, 3)$ *such that*

$$\hat{D}(k+1) \leq (1 - N_1)\hat{D}(k) + N_2\alpha_k^2, \tag{16}$$

$$\bar{D}(k+1) \leq \bar{D}(k) - 2\alpha_k\left(\boldsymbol{z}(k) - f^\star\right) + N_3\alpha_k^2, \tag{17}$$

where $f := -U$, $U^\star$ *is the optimal value of Problem II.1, and* $f^\star := -U^\star$.

Proof: We prove Lemma A.1 by referring to the results in [11]. Assumption II.2 (the Lipschitz continuity of u_i $(i \in \mathcal{V})$) ensures that, for all $i \in \mathcal{V}$, there exists $M_i \in \mathbb{R}$ such that $\sup\{\|\boldsymbol{g}_i\| \colon \boldsymbol{x} \in C_i, \boldsymbol{g}_i \in \partial(-u_i)(\boldsymbol{x})\} \leq M_i$ [5, Theorem 6.2.2, Corollary 6.1.2, and Exercise 6.1.9(c)]. This, together

with Assumptions II.1 and II.2, implies that (A1)–(A3) in [11] hold. Since the assumptions in Lemmas 3.1 and 3.2 in [11] are satisfied, the inequalities in [11, p.42] hold, i.e., there exist[6] $c > 1/6$ and $d > 0$ such that, for all $i \in \mathcal{V}$ and all $k \in \mathbb{N}$,

$$\mathrm{d}(\boldsymbol{x}_i(k+1), C)^2 \leq \mathrm{d}(\boldsymbol{v}_i(k), C)^2 - \frac{1}{6c}\mathrm{d}\left(\boldsymbol{v}_i(k), C\right)^2 + d\alpha_k^2.$$

The convexity of $\mathrm{d}(\cdot, C)^2$ guarantees that $\mathrm{d}(\boldsymbol{v}_i(k), C)^2 \leq \sum_{j\in\mathcal{V}} w_{ij}(k)\mathrm{d}\left(\boldsymbol{x}_j(k), C\right)^2$. Accordingly, Assumption III.3 leads to (16).

Since C is compact, there exists a solution of Problem II.1 [1, Corollary 8.31, Proposition 11.14]. The assumptions in Lemma A.1 and [11, Lemma 3.1] ensure that, for all $i \in \mathcal{V}$ and all $k \in \mathbb{N}$,

$$\begin{aligned}\|\boldsymbol{x}_i(k+1) - \boldsymbol{x}^\star\|^2 &\leq \|\boldsymbol{v}_i(k) - \boldsymbol{x}^\star\|^2 - \frac{1}{6c}\mathrm{d}(\boldsymbol{v}_i(k), C)^2 + d\alpha_k^2 \\ &\quad - 2\alpha_k(f_i(\boldsymbol{z}_i(k)) - f_i(\boldsymbol{x}^\star)) \qquad (18)\\ &\leq \|\boldsymbol{v}_i(k) - \boldsymbol{x}^\star\|^2 + d\alpha_k^2 \\ &\quad - 2\alpha_k(f_i(\boldsymbol{z}_i(k)) - f_i(\boldsymbol{x}^\star)).\end{aligned}$$

An argument similar to the one for obtaining (16) implies (17). This completes the proof. □

Next, we prove Theorem III.1(i).

Proof of Theorem III.1(i): We prove that there exists $M_1 > 0$ such that

$$\liminf_{k\to+\infty} \hat{D}(k) \leq M_1\alpha^2. \qquad (19)$$

To show that $N_1 \liminf_{k\to+\infty} \hat{D}(k) \leq N_2\alpha^2$, we assume that the assertion does not hold, i.e., $N_1 \liminf_{k\to+\infty} \hat{D}(k) > N_2\alpha^2$. Then there exists $\delta > 0$ such that

$$N_1 \liminf_{k\to+\infty} \hat{D}(k) > N_2\alpha^2 + 2\delta.$$

The definition of the limit inferior of $\hat{D}(k)$ means that there exists $k_0 \in \mathbb{N}$ such that, for all $k \geq k_0$,

$$N_1 \liminf_{k\to+\infty} \hat{D}(k) - \delta \leq N_1\hat{D}(k).$$

Accordingly, for all $k \geq k_0$,

$$N_1\hat{D}(k) > N_2\alpha^2 + \delta.$$

Hence, (16) guarantees that, for all $k \geq k_0$,

$$\hat{D}(k+1) \leq \hat{D}(k) - \delta \leq \hat{D}(k_0) - \delta(k+1-k_0),$$

which is a contradiction since the right-hand side of the above inequality approaches minus infinity when k diverges. Therefore, $N_1 \liminf_{k\to+\infty} \hat{D}(k) \leq N_2\alpha^2$, i.e., (19) holds.

We show that, for all $\epsilon > 0$,

$$2\alpha\left(\liminf_{k\to+\infty} \boldsymbol{z}(k) - f^\star\right) \leq N_3\alpha^2 + \epsilon. \qquad (20)$$

Here, let us assume that, for all $\epsilon > 0$, (20) does not hold, i.e., there exists $\epsilon_0 > 0$ such that

$$2\alpha\left(\liminf_{k\to+\infty} \boldsymbol{z}(k) - f^\star\right) > N_3\alpha^2 + \epsilon_0.$$

[6]From [16, p.223] (see also Assumption 2.5 in [11]) and the definition of C_i $(i \in \mathcal{V})$, we can set a large number $c > 0$ such that $\mathrm{d}(\boldsymbol{v}_i(k), C)^2 \leq c\mathrm{d}(\boldsymbol{v}_i(k), C_i)^2$ $(i \in \mathcal{V}, k \in \mathbb{N})$.

The definition of the limit inferior of $\boldsymbol{z}(k)$ guarantees that there exists $k_1 \in \mathbb{N}$ such that, for all $k \geq k_1$,

$$2\alpha\left(\liminf_{k\to+\infty} \boldsymbol{z}(k) - f^\star\right) - \frac{1}{2}\epsilon_0 \leq 2\alpha\left(\boldsymbol{z}(k) - f^\star\right),$$

which implies that, for all $k \geq k_1$,

$$2\alpha\left(\boldsymbol{z}(k) - f^\star\right) > N_3\alpha^2 + \frac{1}{2}\epsilon_0.$$

Accordingly, from (17), for all $k \geq k_1$,

$$\bar{D}(k+1) \leq \bar{D}(k) - \frac{1}{2}\epsilon_0 \leq \bar{D}(k_0) - \frac{1}{2}\epsilon_0(k+1-k_1),$$

which leads to a contradiction since the right-hand side of the above inequality approaches minus infinity when k diverges. Therefore, (20) holds for all $\epsilon > 0$. Since $\epsilon > 0$ is arbitrary, we have

$$2\alpha\left(\liminf_{k\to+\infty} \boldsymbol{z}(k) - f^\star\right) \leq N_3\alpha^2,$$

which implies that

$$\liminf_{k\to+\infty} \boldsymbol{z}(k) - f^\star \leq \frac{N_3}{2}\alpha.$$

This completes the proof of Theorem III.1(i). □

Finally, we prove Theorem III.1(ii).

Proof of Theorem III.1(ii): Theorem 3.1 in [11] guarantees that, under Assumptions (A1)–(A3) in [11], Assumption 2.2 in [11] (which is the same as Assumption III.2 in the present paper), and Assumption 2.3 in [11] (which is the same as Assumption III.3 in the present paper), the sequence $(\boldsymbol{x}_i(k))_{k\in\mathbb{N}}$ $(i \in \mathcal{V})$ generated by algorithm (15) with $P_i = \mathsf{P}_{i,j}$ $(i \in \mathcal{V}, j = 1, 2, \ldots, J_i)$ (i.e., Algorithm 1) converges to a solution of Problem II.1.

Summing (16) from $k = 1$ to $k = K \in \mathbb{N}$ ensures that

$$N_1 \sum_{k=1}^{K} \hat{D}(k) \leq \hat{D}(1) - \hat{D}(K+1) + N_2 \sum_{k=1}^{K} \alpha_k^2,$$

which, together with $\sum_{k=1}^{+\infty} \alpha_k^2 < +\infty$ and the convexity of $\mathrm{d}(\cdot, C)^2$, implies that there exists $N_4 > 0$ such that

$$\sum_{i\in\mathcal{V}} \mathrm{d}\left(\frac{1}{K}\sum_{k=1}^{K} \boldsymbol{x}_i(k), C\right)^2 \leq \frac{1}{K}\sum_{k=1}^{K} \hat{D}(k) \leq \frac{N_4}{K}.$$

Accordingly, (7) holds.

Inequality (18) and the assumption of the existence of $N_5 > 0$ such that $N_5 \sum_{i\in\mathcal{V}} \|\boldsymbol{v}_i(k) - \boldsymbol{x}^\star\|^2 \leq \sum_{i\in\mathcal{V}} \mathrm{d}(\boldsymbol{v}_i(k), C)^2$ $(k \in \mathbb{N})$, together with Assumption III.3, imply that there exists $c > N_5/6$ such that

$$\bar{D}(k+1) \leq (1-p)\bar{D}(k) - 2\alpha_k(\boldsymbol{z}(k) - f^\star) + N_3\alpha_k^2,$$

where $p := N_5/(6c)$ and $N_3 := dV$. In the case where $\alpha_k \in (0, 1]$ $(k \in \mathbb{N})$, for all $k \in \mathbb{N}$,

$$\boldsymbol{z}(k) - f^\star \leq \frac{1-p\alpha_k}{2\alpha_k}\bar{D}(k) - \frac{1}{2\alpha_k}\bar{D}(k+1) + \frac{N_3}{2}\alpha_k. \qquad (21)$$

(a) Let $\alpha_k = 1/(pk)$ $(k \in \mathbb{N})$. Inequality (21) implies that, for all $k \in \mathbb{N}$,

$$\boldsymbol{z}(k) - f^\star \leq \frac{p(k-1)}{2}\bar{D}(k) - \frac{pk}{2}\bar{D}(k+1) + \frac{N_3}{2pk}.$$

Summing the above inequality from $k = 1$ to $k = K \in \mathbb{N}$ ensures that

$$\frac{1}{K}\sum_{k=1}^{K} \boldsymbol{z}(k) - f^\star \leq -\frac{p}{2}\bar{D}(K+1) + \frac{N_3}{2p}\frac{1+\log K}{K},$$

where $\sum_{k=1}^{K}(1/k) \leq 1 + \log K$. The convexity of f_i $(i \in \mathcal{V})$ thus guarantees that

$$\sum_{i\in\mathcal{V}} f_i\left(\frac{1}{K}\sum_{k=1}^{K}\boldsymbol{z}_i(k)\right) \leq f^\star - \frac{p}{2}\bar{D}(K+1) + \frac{N_3}{2p}\frac{1+\log K}{K} \leq f^\star + \frac{N_3}{2p}\frac{1+\log K}{K}.$$

(b) Let $\alpha_k = 2/(p(k+1))$ $(k \in \mathbb{N})$. From (21), for all $k \in \mathbb{N}$,

$$k(\boldsymbol{z}(k) - f^\star) \leq \frac{p(k-1)k}{4}\bar{D}(k) - \frac{pk(k+1)}{4}\bar{D}(k+1) + N_6,$$

where $(N_3 k)/(p(k+1)) \leq N_3/p =: N_6$. Summing the above inequality from $k = 1$ to $k = K \in \mathbb{N}$ ensures that

$$\frac{2}{K(K+1)}\sum_{k=1}^{K} k\boldsymbol{z}(k) - f^\star \leq -\frac{p}{2}\bar{D}(K+1) + \frac{2N_6}{K+1},$$

which, together with the convexity of f_i $(i \in \mathcal{V})$, implies that

$$\sum_{i\in\mathcal{V}} f_i\left(\frac{2}{K(K+1)}\sum_{k=1}^{K} k\boldsymbol{z}_i(k)\right) \leq f^\star - \frac{p}{2}\bar{D}(K+1) + \frac{2N_6}{K} \leq f^\star + \frac{2N_6}{K}.$$

This completes the proof of Theorem III.1(ii). □

The results in [11, Section 4], together with the discussion in Appendix A, imply Theorem III.2. Therefore, we omit the proof of Theorem III.2.

Acknowledgment

I am sincerely grateful to the associate editor, Tansu Alpcan, and the three anonymous reviewers for helping me improve the original manuscript. I also would like to thank Kazuhiro Hishinuma for his input on the numerical examples.

References

[1] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer, New York, 2011.
[2] A. Beck, A. Nedić, A. Ozdaglar, and M. Teboulle, *An $O(1/k)$ gradient method for network resource allocation problems*, IEEE Trans. Control. Network Systems, vol. 1, no. 1, pp. 64–73, 2014.
[3] D. P. Bertsekas, *Incremental proximal methods for large scale convex optimization*, Math. Program. vol. 129, no. 2, pp. 163–195, 2011.
[4] D. P. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, New Jersey, 1987.
[5] J. M. Borwein and A. S. Lewis, *Convex Analysis and Nonlinear Optimization: Theory and Examples*, Springer, New York, 2006.
[6] R. A. Brualdi and H. J. Ryser, *Combinatorial Matrix Theory*, Cambridge University Press, Cambridge, 1991.
[7] P. L. Combettes and J. C. Pesquet, *A Douglas-Rachford splitting approach to nonsmooth convex variational signal recovery*, IEEE J. Sel. Top. Signal Process. vol. 1, no. 4, pp. 564–574, 2007.
[8] P. L. Combettes and J. C. Pesquet, *Proximal splitting methods in signal processing*, in: H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, H. Wolkowicz (Eds.), Fixed-Point Algorithms for Inverse Problems in Science and Engineering, Springer, New York, pp. 185–212, 2011.
[9] D. Easley and J. Kleinberg, *Networks, Crowds, and Markets: Reasoning about a highly connected world*, Cambridge University Press, Cambridge, 2010.
[10] H. Iiduka, *Iterative algorithm for triple-hierarchical constrained nonconvex optimization problem and its application to network bandwidth allocation*, SIAM J. Optim., vol. 22, no. 3, pp. 862–878, 2012.
[11] H. Iiduka, *Almost sure convergence of random projected proximal and subgradient algorithms for distributed nonsmooth convex optimization*, Optimization, vol. 66, no. 1, pp. 35–59, 2017.
[12] H. Iiduka and M. Uchida, *Fixed point optimization algorithms for network bandwidth allocation problems with compoundable constraints*, IEEE Communications Letters, vol. 15, no. 6, pp. 596–598, 2011.
[13] F. Kelly, *Charging and rate control for elastic traffic*, European Trans. Telecommun. vol. 8, no. 1, pp. 33–37, 1997.
[14] D. Kraft, *A software package for sequential quadratic programming*, Tech. Rep. DFVLR-FB 88-28, DLR German Aerospace Center–Institute for Flight Mechanics, Koln, Germany, 1988.
[15] D. Knuth, *Big Omicron and big Omega and big Theta*, ACM SIGACT News, vol. 8, no. 2, pp. 18–24, 1976.
[16] S. Lee and A. Nedić, *Distributed random projection algorithm for convex optimization*, IEEE J. Sel. Top. Signal Process. vol. 7, no. 2, pp. 221–229, 2013.
[17] S. H. Low and D. E. Lapsley, *Optimization flow control–I: Basic algorithm and convergence*, IEEE/ACM Trans. Networking, vol. 7, no. 6, pp. 861–874, 1999.
[18] J. Marašević, C. Stein, and G. Zussman, *A fast distributed stateless algorithm for α-fair packing problems*, Proc. ICALP'16, 2016.
[19] G. J. Minty, *A theorem on maximal monotonic sets in Hilbert space*, J. Math. Anal. Appl. vol. 11, pp. 434–439, 1965.
[20] J. Mo and J. Walrand, *Fair end-to-end window-based congestion control*, IEEE/ACM Trans. Networking, vol. 8, no. 5, pp. 556–567, 2000.
[21] J. J. Moreau, *Fonctions convexes duales et points proximaux dans un espace hilbertien*, C. R. Acad. Sci. Paris Sér. A Math. vol. 255, pp. 2897–2899, 1962.
[22] A. Nedić and D. P. Bertsekas, *Incremental subgradient methods for nondifferentiable optimization*, SIAM J. Optim. vol. 12, no. 1, pp. 109–138, 2001.
[23] A. Nedić and A. Ozdaglar, *Approximate primal solutions and rate analysis for dual subgradient methods*, SIAM J. Optim. vol. 19, no. 4, pp. 1757–1780, 2009.
[24] A. Nedić and A. Ozdaglar, *Subgradient methods for saddle-point problems*, J. Optim. Theory Appl. vol. 142, no. 1, pp. 205–228, 2009.
[25] N. Parikh and S. Boyd, *Proximal Algorithms*, Foundations and Trends in Optimization, vol. 1, no. 3, pp. 127–239, 2014.
[26] R. Srikant, *The Mathematics of Internet Congestion Control*, Birkhäuser, Boston, 2004.
[27] E. Wei, A. Ozdaglar, and A. Jadbabaie, *A distributed Newton method for network utility maximization–I: Algorithm*, IEEE Trans. Automat. Control, vol. 58, no. 9, pp. 2162–2175, 2013.
[28] E. Wei, A. Ozdaglar, and A. Jadbabaie, *A distributed Newton method for network utility maximization–Part II: Convergence*, IEEE Trans. Automat. Control, vol. 58, no. 9, pp. 2176–2188, 2013.
[29] L. Xiao and S. Boyd, *Fast linear iterations for distributed averaging*, Syst. Control Lett. vol. 53, no. pp. 65–78, 2004.
[30] H. Yu and M. J. Neely, *A simple parallel algorithm with an $O(1/t)$ convergence rate for general convex programs*, SIAM J. Optim. vol. 27, no. 2, pp. 759–783, 2017.

Hideaki Iiduka received the Ph.D. degree from Tokyo Institute of Technology, Tokyo, Japan, in 2005. From 2005 to 2007, he was a Research Assistant in the Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Tokyo, Japan. From 2007 to 2008, he was a Research Fellow (PD) of the Japan Society for the Promotion of Science. From October 2008 to March 2013, he was an Associate Professor in the Network Design Research Center, Kyushu Institute of Technology, Tokyo, Japan. Since April 2013, he has been an Associate Professor in the Department of Computer Science, School of Science and Technology, Meiji University, Kanagawa, Japan. His research field is optimization theory and its applications to mathematical information science. He is a member of SIAM and MOS.