Multi-Layer Decomposition of Network Utility Maximization Problems

Nurullah Karakoç, *Graduate Student Member, IEEE*, Anna Scaglione, *Fellow, IEEE*, Angelia Nedić, *Member, IEEE*, and Martin Reisslein, *Fellow, IEEE*

***Abstract*— We describe a distributed framework for resource sharing problems that arise in communications, micro-economics, and various networking applications. In particular, we consider a hierarchical multi-layer decomposition for network utility maximization (ML-NUM), where functionalities are assigned to different layers. The proposed methodology creates solutions with central management and distributed computations to the resource allocation problems. In non-stationary environments, the technique aims to respond quickly to the dynamics of the network by decreasing delay by partially shifting the communication and computational burden to the network edges. Our main contribution is a detailed analysis under the assumption that the network changes are on the same time-scale as the convergence time of the algorithms used for local computations. Moreover, assuming strong concavity and smoothness of the users' objective functions, and under some stability conditions for each layer, we present convergence rates and optimality bounds for the ML-NUM framework. In addition, the main benefits of the proposed method are demonstrated with numerical examples.**

***Index Terms*— Distributed computation, network resource allocation.**

I. Introduction

IN THIS paper, we revisit the classic problem of allocating resources in a network with many users through decomposition into interacting layers with a central mechanism that distributes resources among these different users based on their demand profiles. The formulation was inspired by our recent work on coordination of radio access networks (RANs) using a Software Define Networked (SDN) orchestrator at the network backhaul [2]–[4]. RANs include a devices layer and a layer for radio nodes (e.g., cellular base stations (eNBs) and Wi-Fi access points (APs)), and interact with a layer of gateways (e.g., baseband processing units (BBUs)), as well as additional layers for switches and core network entities. Unlike [2]–[4], this paper focuses on building the theoretical groundwork, formulating the problem in abstract terms and studying in depth its performance. The abstract decomposition we consider finds other applications in, for instance, the optimal distribution of a product among warehouses for several retailers [5], the optimal caching of content on Internet servers [6], or the optimal allocations in *transactive energy* markets [7].

Manuscript received November 12, 2018; revised September 27, 2019 and April 1, 2020; accepted June 10, 2020; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor J. Shin. This work was supported in part by the NSF under Grant CCF-1717391 and Grant NeTS-1716121 and in part by the ONR under Grant N000141612245. This article appears in part in the proceedings of the IEEE Conference on Decision and Control, Miami Beach, FL, USA, Dec. 2018. *(Corresponding author: Nurullah Karakoç.)*

The authors are with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: nkarakoc@asu.edu; anna.scaglione@asu.edu; angelia.nedich@asu.edu; reisslein@asu.edu).

Digital Object Identifier 10.1109/TNET.2020.3003925

In all of these contexts, different layers of intermediate entities exist naturally, due to the structure of the service and of the control framework. It is clear that delegating all the decisions to a central controller is not reasonable. In order to scale, any mechanism for coordination must decompose the resource allocation problem in a way that limits the control signaling overhead between these entities, while at the same time being able to respond rapidly to changes in resources demand or supply.

Resource allocation problems have been extensively studied in the literature, especially under the network utility maximization (NUM) framework, where fair resource allocation to heterogeneous users is formulated as a convex problem [8], [9]. Within the NUM literature, a variety of decentralized solutions (surveyed in [10]) have been presented with different assumptions and setups. These solutions commonly use the method of parameter exchanges between edges and the central control to build a decentralized implementation corresponding to the mathematical decomposition of the problem (see [11], [12] for various decomposition techniques). An important issue in these solutions is the relationship between the time scales of the computations and the demand changes. The common assumption is that the convergence time of the proposed methods is much shorter than the time for changes to occur in the demand; in other words, the demand is assumed to be static during the execution of the methods. This assumption, referred to as time-scale separation, separates the optimization procedure from the demand profile dynamics and works well when the demand is relatively stable. In [13], the stability region of a dual optimal flow control algorithm is studied for the case where the time-scale separation assumption is relaxed such that the number of users changes with the arrival and service rates of a Markovian queueing model.

The time-scale separation assumption can be problematic in real systems, as it may lead to outdated optimal distributions that are not optimal for the current demand. When the network changes are on the same time-scale as the convergence time of the gradient method type algorithms, errors occur in the gradient vectors which lead to perturbations from the ideal trajectory of the iterations. Errors in optimization methods have been the focus of substantial research. Specifically, erroneous gradient methods have been studied extensively. Strong convergence rate results have been characterized in [14] with diminishing step size and Lipschitz gradients for stochastic and deterministic errors. Nedić and Bertsekas have focused on the effect of deterministic noise on subgradient methods in [15], and characterized the convergence properties for various step size selections. The effects of estimation errors on the dual variables in a dual-gradient flow control scheme have been analyzed in [16]. Zhang *et al.* have investigated the impacts of stochastic noisy feedback on distributed NUM in [17] with

a particular focus on primal-dual algorithms with diminishing step size. Also, [18], [19] have studied imperfect parameter passing in distributed algorithms for different applications. When we remove the time-scale separation assumption, the optimal solution changes within time. Some fundamental results have been derived by Popkov in [20] for the asymptotic behavior of gradient methods in non-stationary environments. Xi and Khan have studied distributed dynamic optimization over directed graphs in [21], and presented an algorithm converging linearly to an error bound in non-stationary environments under strong convexity and Lipschitz gradient assumptions for objective functions.

A. Contributions

In this paper, we propose a hierarchical decomposition approach to the resource allocation problem by introducing slack variables and using the idea of clustering nodes at each layer of the hierarchy. The distributed algorithm proposed to solve the resource allocation problem is based on projected gradient iterations. The idea behind the approach is simple: with the slack variables, we create multiple layers between the end-users and the central controller. At the lowest layer are the users which are clustered under the nodes in higher layers; the nodes in a higher layer act as distributors for the lower layer elements in their cluster.

In terms of analysis of our multi-layer algorithm, our technical contributions are as follows. First, we generalize results for the standard two-layer case and prove that the multi-layer algorithm achieves linear convergence rate for constant step size under some concavity and smoothness assumptions on the utility functions and show under what conditions the latency of the multi-layer algorithm is lower than that of the two-layer counterpart (in Sec. III). Second, we provide novel error bounds characterizing the sub-optimality of the solutions produced by the algorithm in dynamic allocation scenarios (in Sec. IV). Finally, in Sec. V, we corroborate the analysis with numerical results that clearly illustrate the benefits of the decomposition and confirm the trends expected from the theoretical study.

II. Problem Formulation

Consider a network of N users, each with the objective to maximize his/her own (scalar) utility function. The users' decisions are coupled through a common resource, whereby the amount of total resource is limited by R_{tot}. We can formulate this resource allocation problem as follows:

$$\max_{x_s \in \mathcal{I}_s} \sum_{s=1}^{N} U_s(x_s) \quad \text{s.t.} \quad \sum_{s=1}^{N} x_s \leq R_{\text{tot}}, \tag{1}$$

where s denotes the user index, x_s is the amount of resource consumed by the sth end-user, $\mathcal{I}_s = [m_s, M_s]$ is the interval representing the local constraints for user s, with $0 \leq m_s < M_s$, and $U_s(x_s)$ is his/her utility function. We denote by $\mathbf{x}$ the vector of users' resources, i.e., $\mathbf{x} = (x_1, x_2, \ldots, x_N)^\top$ where $[\cdot]^\top$ denotes the transpose operation.

Replacing x_s by y_s^1, we define the L-layer decomposition problem as follows:

$$\begin{aligned} &\max_{\mathbf{y}} \ \sum_{s=1}^{N} U_s(y_s^1) \\ &\text{s.t. } y_1^L = R_{\text{tot}}, \quad m_s \leq y_s^1 \leq M_s, \ s = 1, \ldots, N, \\ &\sum_{s \in \mathcal{S}_\ell^n} y_s^\ell \leq y_n^{\ell+1}, \quad n = 1, \ldots, N_{\ell+1}, \ \ell = 1, \ldots, L-1, \end{aligned} \tag{2}$$

which includes $L-2$ sets of slack variables. Here, the end-users (bottom) layer is called layer-1 (corresponding to $\ell = 1$), the central distribution layer (top) is called layer-L (with index $\ell = L$). The scalar y_s^ℓ represents the total resource of the sth node in layer-ℓ for $s = 1, \ldots, N_\ell$, where N_ℓ denotes the number of nodes in layer ℓ. These nodes are partitioned into $N_{\ell+1}$ disjoint sets $\mathcal{S}_\ell^n$ that represent all the nodes that are connected with node n of the layer $\ell+1$ above layer ℓ. Naturally, $N_1 = N$ and $N_L = 1$. We assume that each element from layer ℓ is only connected to one node in layer $\ell+1$. With $\mathbf{y}$ we denote the vector of resources for all layers. Fig. 1 illustrates an example with a 3-layer architecture.

A given node in a higher layer acts as a distributor of resources to the nodes in the corresponding cluster in the lower layer. When these layers have no constraints other than just feasibility relations as in (2), the problems (1) and (2) are equivalent. This decomposition is useful since it naturally reflects modules and boundaries that are present in a communication network architecture. Hence, the computational resources that can tackle the intermediate problems are already in place, they simply currently do not dynamically co-optimize their shares of resources. Also, geographical clustering is natural in network services, and in this case the clusters represent main control points in each region.

In the distributed solutions of problems (1) and (2), each layer is responsible for their local computations, and the coordination among them is carried out by parameter passings between these layers. These computations, in general, are iterative, where upper layer elements in each iteration use the results of the lower layer's multiple iterations due to the nature of the solution procedure. Hence, the iterations in upper layers use slower time-scales since they need to wait for lower layer iteration results. Therefore, the main controller/distributor needs to wait for the transmissions of the parameters from the end-users in the classic setup in (1). However, in multi-layer decomposition, instead of communicating with all the users, the main controller only gathers information from the sub-controllers (intermediate layer entities). Assuming the latency in a link is proportional with the distance and the number of nodes communicating in parallel, by shifting the computational and communication burden to the sub-entities, multi-layer decomposition can reduce the latency, leading to faster iterations at the top layer. In the next section, we introduce distributed algorithms and analyze their convergence properties.

III. Distributed Algorithm and Its Convergence

We make the following assumptions:

a.1) Utility functions U_s are increasing, strictly concave, and twice continuously differentiable on the interval $\mathcal{I}_s = [m_s, M_s]$.

a.2) The curvatures of U_s are bounded away from zero such that $-\ddot{U}_s(x_s) \geq 1/\alpha_s > 0$ for all $x_s \in \mathcal{I}_s$, where $\ddot{U}_s$ denotes the second derivative of U_s.

a.3) A feasible solution exists, i.e., $R_{\text{tot}} \geq \sum_{s=1}^{N} m_s$.

For example, some TCP algorithms considering fair allocation in networks [10], [22] have utility functions belonging to a group of functions parametrized with a scalar $a_s > 0$

$$U_s(x_s) = \begin{cases} w_s \log x_s, & a_s = 1 \\ w_s (1-a_s)^{-1} x_s^{1-a_s}, & a_s \neq 1, \end{cases} \tag{3}$$

where weights $w_s > 0$ and $0 < m_s < M_s$ for all $x_s \in \mathcal{I}_s$. A utility function selected from this group satisfies **a.1** and **a.2**. Another example of the utility functions satisfying these assumptions can be $U_s(x_s) = w_s \log(1 + x_s)$ where $w_s > 0$ and $0 \leq m_s < M_s$. As an additional property, this function results in nonnegative utilities for all $x_s \in \mathcal{I}_s$.

Before discussing our multi-layer decomposition algorithm in Section III-B, we first review the conventional two-layer decomposition method (see e.g., [9]) and its performance guarantees in the next section.

A. 2-Layer Algorithm and Convergence Results

Distributed optimization algorithms for solving problem (1) that come from the dual decomposition are based on writing the Lagrangian function arising from the relaxation of the total resource constraint as a separable problem, i.e.,

$$\begin{aligned} L(\mathbf{x}, \lambda) &= \sum_{s=1}^{N} U_s(x_s) - \lambda\left(\sum_{s=1}^{N} x_s - R_{\text{tot}}\right) \\ &= \sum_{s=1}^{N} \left(U_s(x_s) - \lambda x_s\right) + \lambda R_{\text{tot}}, \end{aligned} \tag{4}$$

where λ is the Lagrange multiplier. The dual objective is:

$$g(\lambda) = \max_{x_s \in \mathcal{I}_s, 1 \leq s \leq N} L(\mathbf{x}, \lambda) = \sum_{s=1}^{N} f_s(\lambda) + \lambda R_{\text{tot}}, \tag{5}$$

where for each user $s = 1, \ldots, N$,

$$f_s(\lambda) = \max_{x_s \in \mathcal{I}_s} \{U_s(x_s) - \lambda x_s\} \tag{6}$$

and the dual problem is

$$\min_{\lambda \geq 0} g(\lambda). \tag{7}$$

The decentralized implementation finds the optimal resource allocation x_s of the subproblems in (6) for a given Lagrange multiplier λ, while the optimum value of λ in (7) is found via a gradient projection algorithm. Specifically, the iterations for λ are:

$$\begin{aligned} \lambda(t+1) &= \left[\lambda(t) - \gamma \frac{\partial g(\lambda(t))}{\partial \lambda}\right]^{+} \\ &= \left[\lambda(t) + \gamma\left(\sum_{s=1}^{N} x_s^*(\lambda(t)) - R_{\text{tot}}\right)\right]^{+}, \end{aligned} \tag{8}$$

where $\gamma > 0$ denotes the step size and $[\cdot]^+$ represents the projection onto the nonnegative orthant. Furthermore, $x_s^*(\lambda)$ denotes the solution to subproblem (6) for a given λ, i.e.,

$$x_s^*(\lambda) = [\dot{U}_s^{-1}(\lambda)]_{m_s}^{M_s}, \tag{9}$$

where $\dot{U}_s^{-1}$ is the inverse function of $\dot{U}_s$ which denotes the derivative of the utility function for end-user s, and $[\cdot]_{m_s}^{M_s}$ is the projection onto set $\mathcal{I}_s$, i.e., $[z]_a^b = \min(\max(z, a), b)$.

In a nutshell, the algorithm works as follows. Each user updates optimal resources $x_s^*(\lambda)$ for a given λ according to (9) and passes this value to the central controller. Then, the controller updates the price λ according to (8) and passes this value to the users. This process continues iteratively until some convergence criterion is satisfied. The typical analogy is the law of supply and demand from economics. Each user gets resources with *price* λ with cost λx_s. Then, each user maximizes the utility function minus cost for a given price λ and decides on his/her optimal resource allocation $x_s(\lambda)$. The central distributor, on the other hand, decides on the optimal price λ^* by increasing or decreasing it according to the total supply R_{tot} and demand $\sum_{s=1}^{N} x_s$.

The convergence of the algorithm is shown in [9, Thm. 1] with a different problem setup which includes link constraints without specifying the convergence rate. Note that, by using (4) and (7), we can find the second derivative of the dual objective function $g(\lambda)$:

$$\frac{\partial^2 g(\lambda)}{\partial \lambda^2} = -\sum_{s=1}^{N} \frac{\partial x_s^*(\lambda)}{\partial \lambda}, \tag{10}$$

where, from (9), we have

$$\frac{\partial x_s^*(\lambda)}{\partial \lambda} = \begin{cases} \dfrac{1}{\ddot{U}_s(x_s^*(\lambda))}, & \text{for } \dot{U}_s(m_s) \geq \lambda \geq \dot{U}_s(M_s), \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

From **a.2**, it follows that

$$0 < -\frac{1}{\ddot{U}_s(x_s^*)} \leq \alpha_s < \infty \quad \text{for all } s. \tag{12}$$

Next, we define $\overline{\alpha} = \max_s(\alpha_s)$. Under assumptions **a.1** and **a.2**, following the analysis that is similar to that of [9], it can be shown that $g(\lambda)$ has Lipschitz gradients with a constant $N\overline{\alpha}$. Under assumptions **a.1**–**a.3**, the dual problem (7) has a nonempty solution set Λ^*, and we have:

Theorem 1: Under assumptions **a.1**–**a.3**, the method (8) with a constant step size $0 < \gamma \leq 1/(N\overline{\alpha})$ produces iterates such that $\lim_{t\to\infty} g(\lambda(t)) = g^*$, and its convergence rate is sub-linear with $\mathcal{O}(1/t)$:

$$g(\lambda(t)) - g^* \leq \frac{1}{2\gamma t} \left\|\lambda(0) - \lambda^*\right\|^2, \quad \forall \lambda^* \in \Lambda^*, t \geq 0,$$

where g^* denotes the dual optimal value and $\lambda(0) \geq 0$ is the initial iterate value.[1]

Since a larger step size makes the bound smaller, given the condition $0 < \gamma \leq 1/(N\overline{\alpha})$, the best choice is $\gamma = 1/(N\overline{\alpha})$. With this selection, we obtain

$$g(\lambda(t)) - g^* \leq \frac{N\overline{\alpha}}{2t} \left\|\lambda(0) - \lambda^*\right\|^2. \tag{13}$$

Thus, if we are interested in an $\epsilon = g(\lambda(t)) - g^*$ approximate solution, then we need a number t of iterations to satisfy $t \geq \frac{N\overline{\alpha}(d(0))^2}{2\epsilon}$, where $d(0) = \|\lambda(0) - \lambda^*\|$ for a $\lambda^* \in \Lambda^*$.

Corollary 1: The primal variables in (9), i.e., the allocations $\mathbf{x}^*(\lambda) = [x_1^*(\lambda), \ldots, x_N^*(\lambda)]^\top$ converge to the optimal allocations $\mathbf{x}^*(\lambda^*) = [x_1^*(\lambda^*), \ldots, x_N^*(\lambda^*)]^\top$ with $\mathcal{O}(1/\sqrt{t})$:

$$\left\|\mathbf{x}^*(\lambda(t)) - \mathbf{x}^*(\lambda^*)\right\| \leq \frac{\sqrt{N\overline{\alpha}}\left\|\lambda(0) - \lambda^*\right\|}{\sqrt{t}}. \tag{14}$$

In addition, constraint violations diminish with $\mathcal{O}(1/\sqrt{t})$:

$$\left\|\sum_{s=1}^{N} x_s^*(\lambda(t)) - R_{\text{tot}}\right\| \leq \frac{N\overline{\alpha}\left\|\lambda(0) - \lambda^*\right\|}{\sqrt{t}}. \tag{15}$$

The derivation of this corollary is given in Appendix A.[2]

[1]Here, we modify the convergence theorem in [9, Thm. 1] to get a convergence rate by using [23, Thm. 3.1].

[2]Similar convergence rates are obtained for conic convex problems with dual methods in [24], where faster rates can be achieved by using various averaging schemes [24], [25].

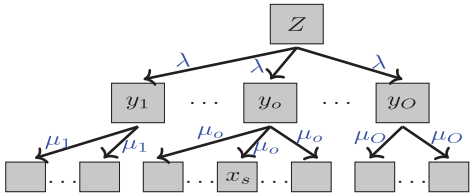


Fig. 1. 3-Layer Decomposition Structure: x_s denotes the allocated resources to user s, and y_o denotes the resources of operator o.

B. A 3-Layer Distributed Algorithm and Convergence Results

We introduce the multi-layer decomposition algorithm with the simplest case. Assume we have the 3-layer decomposition structure for a resource sharing problem as in Fig. 1. Note that for the general multi-layer case, we adopt the notation introduced in Sec. II, specifically in (2). However, for the specific 3-layer case, as shown in Fig. 1, to avoid using superscripts we use x_s, y_o, Z instead of y_s^1, y_o^2, y_1^3, respectively, and μ_o and λ, as the dual variables, which later will be denoted as λ_o^1 and λ_1^2 in the general multi-layer case in Sec. III-C.

In the architecture, we have one controller on the top layer which has a fixed total resource $R_{\text{tot}} = Z$ to share. In the middle layer, we have O nodes, which we will refer to as the *operators*. The resource allocated to the operator with index $o = 1, 2, \ldots, O$, is denoted by y_o. In the bottom layer, we have end-users connected to operators, whereby $\mathcal{S}^o$ represents the set of end-users connected to operator o. To highlight the differences among the intermediate layer and the top and bottom layers, we will refer to $y_1^L = R_{\text{tot}} = Z$ for the top layer and use directly x_s rather than y_s^1 for the bottom layer, while for the 2nd (intermediate) layer we will omit the layer superscript $\ell = 2$ and simply refer to the variables as y_o. More specifically, the optimization problem becomes:

$$\max_{\mathbf{x}\in\mathcal{I},\mathbf{y}\geq\mathbf{0}} \sum_{o=1}^{O}\sum_{s\in\mathcal{S}^o} U_s(x_s)$$
$$\text{s.t.} \sum_{o=1}^{O} y_o \leq Z, \quad \sum_{s\in\mathcal{S}^o} x_s \leq y_o, \quad o = 1,\ldots,O, \quad (16)$$

where $\mathbf{x}, \mathbf{y}$ denote the resources of all nodes in the bottom layer and middle layer, respectively. Here, $\mathcal{I}$ is the Cartesian product of the feasibility intervals $\mathcal{I}_s = [m_s, M_s]$. Relaxing the constraints, we can write the Lagrangian as:

$$L(\mathbf{x},\mathbf{y},\boldsymbol{\mu},\lambda) = \sum_{o=1}^{O}\sum_{s\in\mathcal{S}^o} U_s(x_s) - \sum_{o=1}^{O}\mu_o\Big(\sum_{s\in\mathcal{S}^o} x_s - y_o\Big) - \lambda\Big(\sum_{o=1}^{O} y_o - Z\Big), \quad (17)$$

where $\boldsymbol{\mu}$ is the column vector with entries μ_o for $o = 1,\ldots,O$. Moreover, μ_o is the Lagrange multiplier associated with the feasibility constraint $\sum_{s\in\mathcal{S}^o} x_s \leq y_o$ for each $o = 1,\ldots,O$, which couples layers 1 and 2. The Lagrange multiplier λ is associated with the feasibility constraint $\sum_{o=1}^{O} y_o \leq Z$, coupling layers 2 and 3.

We can write the objective of the dual problem as:

$$g(\lambda,\boldsymbol{\mu}) = \max_{\mathbf{x}\in I,\mathbf{y}\geq 0} L(\mathbf{x},\mathbf{y},\boldsymbol{\mu},\lambda) = \sum_{o=1}^{O}\sum_{s\in\mathcal{S}^o} f_s(\mu_o) + \sum_{o=1}^{O} h_o(\mu_o,\lambda) + \lambda Z, \quad (18)$$

where

$$f_s(\mu_o) = \max_{x_s\in\mathcal{I}_s}\{U_s(x_s) - \mu_o x_s\}, \quad (19)$$
$$h_o(\mu_o,\lambda) = \max_{y_o\geq 0} y_o(\mu_o - \lambda), \quad (20)$$

and the dual problem is

$$\min_{\lambda\geq 0,\boldsymbol{\mu}\geq 0} g(\lambda,\boldsymbol{\mu}). \quad (21)$$

Let $\boldsymbol{\theta} = (\mathbf{x}^\top, \mathbf{y}^\top, \boldsymbol{\mu}^\top, \lambda)^\top$. We can rewrite the dual problem in (21) as

$$\min_{\lambda\geq 0}\min_{\boldsymbol{\mu}\geq 0}\max_{\mathbf{y}\geq 0}\max_{\mathbf{x}\in\mathcal{I}} L(\boldsymbol{\theta}) = \min_{\lambda\geq 0}\max_{\mathbf{y}\geq 0}\min_{\boldsymbol{\mu}\geq 0}\max_{\mathbf{x}\in\mathcal{I}} L(\boldsymbol{\theta}), \quad (22)$$

where we use the separability over variables of both the primal and dual problems in the first step and the minimax theorem[3] to exchange the order of the middle two optimizations in the second step. With this substitution, we obtain an optimization order from the bottom layer to the top layer, as shown in Fig. 1. This optimization order leads to local resource allocation subproblems for the operators that can also be solved in a distributed manner.

1) Distributed Solution: In order to describe the algorithm updates, it is important to notice that in general, multiple iterations are necessary to solve the optimum resource allocation for any given set of values of the dual variables that are associated with the constraints. Let us assume that there are k updates of the dual variables μ_o before a new update of the resource variable y_o, and that y_o is updated k' times before the global price λ is updated. We can define the state of the optimization $\boldsymbol{\theta}(t)$ as follows:

$$\boldsymbol{\theta}(t) = \big(\mathbf{x}^\top(t), \mathbf{y}^\top(\lfloor t/k\rfloor), \boldsymbol{\mu}^\top(t), \lambda(\lfloor t/kk'\rfloor)\big)^\top, \quad (23)$$

where $\lfloor\cdot\rfloor$ denotes the floor function.

In analyzing the problem, we also assume that not all of the users' optimal allocations (as solution of (19) for $s \in \mathcal{S}^o$) are at the boundary points:

a.4) The sets $\mathcal{C}_1^o = \{s \in \mathcal{S}^o : \dot{U}_s(M_s) < \mu_o^*(\lambda,\mathbf{y}) < \dot{U}_s(m_s)\}$ are not empty for $o = 1,\ldots,O$, where $\mu_o^*(\lambda,\mathbf{y})$ denotes the optimal solution of $\min_{\boldsymbol{\mu}\geq 0}\max_{\mathbf{x}\in\mathcal{I}} L(\boldsymbol{\theta})$ for fixed λ and $\mathbf{y}$.

Based on **a.1**-**a.4**, we can write the update rules, for every t:

$$x_s(t) = [\dot{U}_s^{-1}(\mu_o(t))]_{m_s}^{M_s}, \quad (24)$$

$$\mu_o(t+1) = \Big[\mu_o(t) - \gamma_o''\frac{\partial L(\boldsymbol{\theta}(t))}{\partial\mu_o}\Big]_{\underline{\mu}_o}^{\overline{\mu}_o} = \Big[\mu_o(t) + \gamma_o''\Big(\sum_{s\in\mathcal{S}_o} x_s(t) - y_o\big(\lfloor t/k\rfloor\big)\Big)\Big]_{\underline{\mu}_o}^{\overline{\mu}_o}, \quad (25)$$

and for every integer i when $t = ik$:

$$y_o(i+1) = \Big[y_o(i) + \gamma_o'\frac{\partial L(\boldsymbol{\theta}(ik))}{\partial\mu_o}\Big]_{\underline{y}_o}^{\overline{y}_o} = \big[y_o(i) + \gamma_o'\big(\mu_o(ik) - \lambda(\lfloor i/k'\rfloor)\big)\big]_{\underline{y}_o}^{\overline{y}_o}, \quad (26)$$

[3]We use compact convex sets as domains of the optimization variables by introducing projection boundaries later.

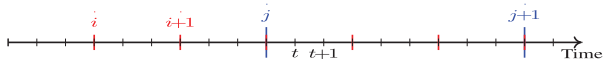


Fig. 2. Time-scale of updates for different iterations. The increasingly coarser time indices t, i, and j index the updates of the local resource prices $\boldsymbol{\mu}$, the operator resources $\mathbf{y}$, and the global price λ, respectively.

and every integer j when $t = jkk'$:

$$\lambda(j+1) = \left[\lambda(j) - \gamma \frac{\partial L(\boldsymbol{\theta}(jkk'))}{\partial \lambda}\right]_{\underline{\lambda}}^{\overline{\lambda}} = \left[\lambda(j) + \gamma\Big(\sum_{o=1}^{O} y_o(jk') - Z\Big)\right]_{\underline{\lambda}}^{\overline{\lambda}}, \tag{27}$$

where $\overline{\mu}_o = \max_{s\in\mathcal{S}^o} \dot{U}_s(m_s)$, $\underline{\mu}_o = \min_{s\in\mathcal{S}^o} \dot{U}_s(M_s)$, $\overline{y}_o = \sum_{s\in\mathcal{S}^o} M_s$, $\underline{y}_o = \sum_{s\in\mathcal{S}^o} m_s$, $\overline{\lambda} = \min_o \overline{\mu}_o$, and $\underline{\lambda} = \max_o \underline{\mu}_o$. These projection boundaries are implied by assumption **a.4** and the feasibility constraints. Also, $\gamma, \gamma'_o, \gamma''_o > 0$ represent different constant step sizes.

Overall, the algorithm ideally works as follows. In the bottom layer, the end-users find the optimal amount of resources x_s^* for a fixed local price μ_o, a fixed operator resource supply y_o, and a fixed global price λ, by using (24). The calculated optimal resource values are then passed to the operators they connect to. Then, the operator o sets a new local price μ_o based on its total supply y_o and the optimal resources x_s requested from the nodes in the lower layer, according to (25). These exchanges continue until the operator comes sufficiently close to the optimum local price μ_o^* for the fixed values of y_o and λ. Then, by using the optimum local prices and general market price λ, the operator iterates the calculation of its resource supply amount y_o according to (26) and uses the iteration outcome to calculate a new optimum local price μ_o^*. This continues until convergence to a value close to the optimum y_o^* for the given global price λ. Then, similarly, by using these operator resource request values y_o^* and the total resource supply Z, the top layer updates the global price λ, at which point all the iterations by the lower layers are repeated with this new λ. The parameter exchanges and iterations continue until some convergence criterion is satisfied.

Extending the analogy with the law of supply and demand from economics, each distribution network connected to an operator with total resource y_o represents a local market establishing a local price μ_o. These operators are connected to the main distributor which represents the global market. The multiplier λ represents the global market price and, at convergence, the local prices coincide with the global price, unless there are specific constraints for the middle layer other than those that come from the decomposition. Fig. 2 shows the different time indices of the different iterations in a toy example[4] where $k = k' = 3$. Section IV focuses on establishing "how fast is fast enough" for the updates in these timescales in order to guarantee the stable performance of the algorithm.

2) Latency Gain: To measure the benefits of the multi-layer design in terms of response time, we compare a two-layer architecture where the N users interact directly with the central controller sharing a control channel c, with a three-layer system where N users are equally split under the control of O operators (i.e., each operator has the same number of users) and the operators use the same channel c of the two-layer system to communicate to the central controller, while the end users use a faster control channel c' (as operators are closer to the users) to interact with their respective operators. To carry out the comparison without entering into the specifics of how the control channel is realized, we can reasonably assume that for a control channel of fixed total capacity, the average delay grows at a linear rate, ρ for channel c and ρ' for channel c', respectively, with respect to the number of sources that access the control channel; naturally, for the faster channel c' the delay growth rate $\rho' < \rho$. Hence, in the two-layer system, the latency for each update of the central controller is ρN. In the three-layer system, between any two iterations in the top layer, there are k' iterations of y_o, however, the operators pass only the last iterate value using the control channel c once. In addition, there are kk' iterations at the bottom end-users which report every iterate value to their respective operators, each with average latency $\frac{N}{O}\rho'$. Therefore the total latency becomes $O\rho + kk'\frac{N}{O}\rho'$. Thus, we gain in latency per iteration if the ratio δ of these two numbers exceeds one, i.e., if:

$$\delta \triangleq \frac{N}{O + kk'\frac{N}{O}\frac{\rho'}{\rho}} > 1. \tag{28}$$

The condition $\delta > 1$ is equivalent to:

$$\frac{kk'}{O}\frac{\rho'}{\rho} < 1 - \frac{O}{N}. \tag{29}$$

We notice that, if the number of operators O is relatively small compared to the number of users N, and if kk' is in the order of O, we have a guaranteed advantage in terms of the latency gain. Alternatively, if we choose $\frac{O}{N} = \frac{\rho'}{\rho}$ we notice that a latency gain requires $kk' < N(1-\rho'/\rho)$; if $\frac{kk'}{N} \ll \frac{\rho'}{\rho}$ then $\delta \approx \rho/\rho'$, showing that for a particularly large population N it is easily advantageous to layer the system.

3) Convergence Rates: For the convergence analysis, in addition to assumptions **a1**–**a3**, we also assume that the curvatures of U_s are bounded from above:

a.5) $$-\ddot{U}_s(x_s) \le 1/\beta_s < \infty \quad \text{for all } x_s \in \mathcal{I}_s. \tag{30}$$

Let:

$$\alpha_o^{(1)} = |\mathcal{S}^o| \max_{s\in\mathcal{S}^o}(\alpha_s), \quad \beta_o^{(1)} = \min_{s\in\mathcal{S}^o}(\beta_s), \qquad \alpha^{(2)} = O \max_o(\alpha_o^{(1)}), \quad \beta^{(2)} = O \min_o(\beta_o^{(1)}), \tag{31}$$

where $|\cdot|$ denotes the cardinality of a set. Assuming the problem has a nonempty solution set, we obtain:

Theorem 2: Let **a.1**–**a.5** hold. For fixed $\lambda(j)$ and $y_o(i)$, the method (25) with constant step size $\gamma''_o = \frac{2}{\alpha_o^{(1)}+\beta_o^{(1)}}$ produces iterates $\mu_o(t)$ that converge to the optimal solution $\mu_o^* := \mu_o^*(\lambda, y_o)$ with a linear rate, i.e., for each $o = 1, \ldots, O$:

$$\|\mu_o(t) - \mu_o^*\| \le \left(\frac{\alpha_o^{(1)} - \beta_o^{(1)}}{\alpha_o^{(1)} + \beta_o^{(1)}}\right)^t \|\mu_o(0) - \mu_o^*\|. \tag{32}$$

Assuming that $\mu_o(t)$ converges to μ_o^* in the bottom layer iterations (i.e., assuming k is large enough), for a fixed $\lambda(j)$, the method (26) with constant step size $\gamma'_o = \frac{2\alpha_o^{(1)}\beta_o^{(1)}}{\alpha_o^{(1)}+\beta_o^{(1)}}$

[4]The selection of $k = k' = 3$ is for illustration purposes only. We note that, in general, each layer needs more iterations to get sufficiently close to the optimal points.

produces iterates $y_o(i)$ converging to the optimal solution $y_o^* := y_o^*(\lambda)$ with a linear rate, i.e., for each $o = 1, \ldots, O$:

$$\|y_o(i) - y_o^*\| \leq \left(\frac{\alpha_o^{(1)} - \beta_o^{(1)}}{\alpha_o^{(1)} + \beta_o^{(1)}}\right)^i \|y_o(0) - y_o^*\|. \quad (33)$$

Assuming that $y_o(i)$ converges to y_o^* in the middle layer iterations (i.e., assuming k' is large enough), the method (27) with constant step size $\gamma = \frac{2}{\alpha^{(2)}+\beta^{(2)}}$ produces iterates $\lambda(j)$ that converge to the optimal solution λ^* with a linear rate, i.e.,

$$\|\lambda(j) - \lambda^*\| \leq \left(\frac{\alpha^{(2)} - \beta^{(2)}}{\alpha^{(2)} + \beta^{(2)}}\right)^j \|\lambda(0) - \lambda^*\|. \quad (34)$$

Proof: See Appendix B. ∎

Note that k and k' do not have a direct effect on the rates of convergence. In fact, Thm. 2 characterizes the convergence rates for each layer in an ideal scenario where upper layer values are fixed and lower layer values have converged. This is equivalent to considering k and k' large enough so that there is no error accumulation from the lower layer iterations. For finite (and possibly small) k and k' values, the algorithms are guaranteed to converge asymptotically to the neighborhoods of the optimal values and the error bounds in Sec. IV specify the error range.

We observe that, when lower layers converge, the convergence rate of the 3-layer algorithm's top layer iterations is the same as that of the 2-layer algorithm convergence rate[5] under assumptions **a.1**–**a.5**. Therefore, if the numbers of lower layer iterations k and k' are sufficiently large, the latency gain characterized in (28) becomes the ratio between the convergence times of these two algorithms.

C. Extension to L-Layer Decomposition

As stated earlier, for the generalized L-layer decomposition, we follow the notation introduced in Sec. II, specifically in (2). Let $\boldsymbol{y}_\ell = (y_1^\ell, \ldots, y_{N_\ell}^\ell)^\top$ be the resources vector for $\ell = 1, 2, \ldots, L$, $U(\boldsymbol{y}_1) = \sum_{s=1}^N U_s(y_s^1)$ be the sum-utility, and the $N_{\ell+1} \times N_\ell$ matrices $\mathbf{A}_\ell$ be the selection matrices whose element (n, s) is 1 for a node s in layer ℓ if $s \in \mathcal{S}_\ell^n$ and 0 otherwise. Introducing the vectors of dual variables for each of the nodes in each of the layers $\boldsymbol{\lambda}_\ell = (\lambda_1^\ell, \ldots, \lambda_{N_{\ell+1}}^\ell)^\top$ (which denote dual variables associated with the ℓth layers' resource constraints $\mathbf{A}_\ell \boldsymbol{y}_\ell \leq \boldsymbol{y}_{\ell+1}$), and:

$$\boldsymbol{\theta} = \begin{pmatrix} \mathbf{y} \\ \boldsymbol{\lambda} \end{pmatrix}, \quad (35)$$

where $\boldsymbol{\lambda}$ denotes the vector of all Lagrange multipliers in all layers, the Lagrangian of problem (2) can be written as:

$$\begin{aligned} L(\boldsymbol{\theta}) &= U(\boldsymbol{y}_1) - \sum_{\ell=1}^{L-1} (\mathbf{A}_\ell \boldsymbol{y}_\ell - \boldsymbol{y}_{\ell+1})^\top \boldsymbol{\lambda}_\ell \\ &= U(\boldsymbol{y}_1) - \boldsymbol{y}_1^\top \mathbf{A}_1^\top \boldsymbol{\lambda}_1 \\ &\quad + \sum_{\ell=2}^{L-1} \boldsymbol{y}_\ell^\top (\boldsymbol{\lambda}_{\ell-1} - \mathbf{A}_\ell^\top \boldsymbol{\lambda}_\ell) + \boldsymbol{\lambda}_{L-1} \boldsymbol{y}_L, \end{aligned} \quad (36)$$

where $\boldsymbol{\lambda}_{L-1}$ and $\boldsymbol{y}_L$ are scalars since there is only one element in the top layer. The objective of the dual problem can be written as:

$$g(\boldsymbol{\lambda}) = f(\boldsymbol{\lambda}_1) + \sum_{\ell=2}^{L-1} h_\ell(\boldsymbol{\lambda}_{\ell-1}, \boldsymbol{\lambda}_\ell) + \boldsymbol{\lambda}_{L-1} \boldsymbol{y}_L, \quad (37)$$

[5]It can be derived similarly by using relation (87) in Appendix B.

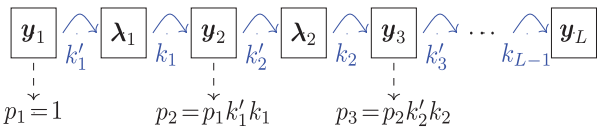


Fig. 3. Decimation factors of time-scales.

where:

$$f(\boldsymbol{\lambda}_1) = \max_{\boldsymbol{y}_1 \in \mathcal{I}} \big(U(\boldsymbol{y}_1) - \boldsymbol{y}_1^\top \mathbf{A}_1^\top \boldsymbol{\lambda}_1\big), \quad (38)$$

$$h_\ell(\boldsymbol{\lambda}_{\ell-1}, \boldsymbol{\lambda}_\ell) = \max_{\boldsymbol{y}_\ell \geq \mathbf{0}} \big(\boldsymbol{y}_\ell^\top (\boldsymbol{\lambda}_{\ell-1} - \mathbf{A}_\ell^\top \boldsymbol{\lambda}_\ell)\big). \quad (39)$$

For the layers $\ell = 1, 2, \ldots, L-1$, there are k_ℓ' updates of the layer resource variables $\boldsymbol{y}_\ell$ for every update of the dual variable $\boldsymbol{\lambda}_\ell$, and there are also k_ℓ updates of the dual variables $\boldsymbol{\lambda}_\ell$ for every $\boldsymbol{y}_{\ell+1}$. Therefore, as shown in Fig. 3, with respect to the index of the fastest updates t at the bottom layer, at each successive higher layer, the number of updates of the resources is decimated by a factor $p_{\ell+1} = k_\ell' k_\ell p_\ell$ and the number of updates of the dual variables is decimated by $k_\ell' p_\ell$, with $p_1 = k_1' = 1$. Once again, we can define the state of the algorithm at time index t as

$$\boldsymbol{\theta}(t) = \begin{pmatrix} \big[\boldsymbol{y}_1(t), \boldsymbol{y}_2(\lfloor t/p_2 \rfloor), \ldots, \boldsymbol{y}_{L-1}(\lfloor t/p_{L-1} \rfloor)\big]^\top \\ \big[\boldsymbol{\lambda}_1(t), \boldsymbol{\lambda}_2(\lfloor t/p_2 k_2' \rfloor), \ldots, \boldsymbol{\lambda}_{L-1}(\lfloor t/p_{L-1} k_{L-1}' \rfloor)\big]^\top \end{pmatrix}. \quad (40)$$

Specifically, the iterations for the dual variables associated with the ℓth layers' resource constraints ($\mathbf{A}_\ell \boldsymbol{y}_\ell \leq \boldsymbol{y}_{\ell+1}$) can be written as a function of the iteration index i as follows:

$$\begin{aligned} \boldsymbol{\lambda}_\ell(i+1) &= \left[\boldsymbol{\lambda}_\ell(i) - \mathbf{G}_\ell \frac{\partial L(\boldsymbol{\theta}(i p_\ell k_\ell'))}{\partial \boldsymbol{\lambda}_\ell}\right]_{\underline{\boldsymbol{\lambda}}_\ell}^{\overline{\boldsymbol{\lambda}}_\ell} \\ &= \big[\boldsymbol{\lambda}_\ell(i) + \mathbf{G}_\ell\big(\mathbf{A}_\ell \boldsymbol{y}_\ell(i k_\ell') - \boldsymbol{y}_{\ell+1}(\lfloor i/k_\ell \rfloor)\big)\big]_{\underline{\boldsymbol{\lambda}}_\ell}^{\overline{\boldsymbol{\lambda}}_\ell}, \end{aligned} \quad (41)$$

where $\mathbf{G}_\ell$ is an $N_{\ell+1} \times N_{\ell+1}$ diagonal matrix of step sizes with $\ell = 1, \ldots, L-1$. The iterations performed to update the values of the resource allocations $\boldsymbol{y}_\ell$ correspond to the recursions:

$$\begin{aligned} \boldsymbol{y}_\ell(j+1) &= \left[\boldsymbol{y}_\ell(j) + \mathbf{G}_\ell' \frac{\partial L(\boldsymbol{\theta}(j p_\ell))}{\partial \boldsymbol{y}_\ell}\right]_{\underline{\boldsymbol{y}}_\ell}^{\overline{\boldsymbol{y}}_\ell} \\ &= \big[\boldsymbol{y}_\ell(j) + \mathbf{G}_\ell'\big(\boldsymbol{\lambda}_{\ell-1}(j k_{\ell-1}) - \mathbf{A}_\ell^\top \boldsymbol{\lambda}_\ell(\lfloor j/k_\ell' \rfloor)\big)\big]_{\underline{\boldsymbol{y}}_\ell}^{\overline{\boldsymbol{y}}_\ell}, \end{aligned} \quad (42)$$

where $\mathbf{G}_\ell'$ is an $N_\ell \times N_\ell$ diagonal matrix of step sizes with $\ell = 2, \ldots, L-1$, and for $\ell = 1$, we have

$$y_s^1(t) = [\dot{U}_s^{-1}(\lambda_n^1(t))]_{m_s}^{M_s}, \quad s \in \mathcal{S}_1^n. \quad (43)$$

Here, we can specify the projection boundaries with recursive relations $\overline{\lambda}_n^\ell = \min_{s \in \mathcal{S}_\ell^n} \overline{\lambda}_s^{\ell-1}$ and $\underline{\lambda}_n^\ell = \max_{s \in \mathcal{S}_\ell^n} \underline{\lambda}_s^{\ell-1}$ for $\ell = 2, 3, \ldots, L-1$, where $\overline{\lambda}_n^1 = \max_{s \in \mathcal{S}_1^n} \dot{U}_s(m_s)$ and $\underline{\lambda}_n^1 = \min_{s \in \mathcal{S}_1^n} \dot{U}_s(M_s)$. Similarly, $\overline{y}_n^\ell = \sum_{s \in \mathcal{S}_{\ell-1}^n} \overline{y}_s^{\ell-1}$ and $\underline{y}_n^\ell = \sum_{s \in \mathcal{S}_{\ell-1}^n} \underline{y}_s^{\ell-1}$ for $\ell = 3, \ldots, L-1$, where $\overline{y}_n^2 = \sum_{s \in \mathcal{S}_1^n} M_s$ and $\underline{y}_n^2 = \sum_{s \in \mathcal{S}_1^n} m_s$. We summarize the algorithmic steps as Algorithm 1.

The result of Theorem 2 can be extended to the L-layer decomposition for problem (2). There is a pattern in the convergence analysis that allows us to obtain the convergence

Algorithm 1 ML-NUM

-At agent s of layer 1 (bottom layer), $\forall s \in \mathcal{S}_2^n$:

At times $t = 1, 2, \ldots$

Update y_s^1 with (43) and report it to n of upper layer

-At agent n of layer ℓ, $\forall n \in \mathcal{S}_\ell^o$, $\forall \ell = 2, \ldots, L-1$:

When t mod $(k'_{\ell-1}p_{\ell-1}) = 0$

1. Receive $y_s^{\ell-1}$'s from the lower layer, $\forall s \in \mathcal{S}_{\ell-1}^n$
2. Update $\lambda_n^{\ell-1}$ with (41) and send it to all $s \in \mathcal{S}_{\ell-1}^n$

When t mod $(p_\ell) = 0$

Update y_n^ℓ with (42)

When t mod $(k'_\ell p_\ell) = 0$

Report y_n^ℓ to agent o in the upper layer

-At the agent of layer L (top layer):

When t mod $(k'_{L-1}p_{L-1}) = 0$

1. Receive all y_n^{L-1}'s from the lower layer elements,
2. Update $\boldsymbol{\lambda}_{L-1}$ with (41) and send it to layer $L-1$

rates of all layers for both dual variable updates and allocated resource updates, similar to 3-layer case (see the details in Appendix C).

We can introduce a recursive definition of:

$$\alpha_n^{(\ell)} = |\mathcal{S}_\ell^n| \max_{s\in\mathcal{S}_\ell^n}(\alpha_s^{(\ell-1)}), \quad \beta_n^{(\ell)} = |\mathcal{S}_\ell^n| \min_{s\in\mathcal{S}_\ell^n}(\beta_s^{(\ell-1)}), \quad (44)$$

for $\ell = 2, 3, \ldots, L-1$, where $\alpha_n^{(1)} = |\mathcal{S}_1^n| \max_{s\in\mathcal{S}_1^n}(\alpha_s)$ and $\beta_n^{(1)} = \min_{s\in\mathcal{S}_1^n}(\beta_s)$. With respect to the dual variables λ_n^ℓ associated with the constraints of the ℓth layers' resources $\sum_{s\in\mathcal{S}_\ell^n} y_s^\ell \le y_n^{\ell+1}$ for $n = 1, \ldots, N_{\ell+1}$, when the upper layer parameters are fixed and assuming that the optimal value $y_s^{\ell*}(\lambda_n^\ell)$ is reached for any λ_n^ℓ, the projected gradient method converges to the optimum solution $\lambda_n^{\ell*}$ with a linear rate, i.e., for $n = 1, \ldots, N_{\ell+1}$:

$$\left\|\lambda_n^\ell(i) - \lambda_n^{\ell*}\right\| \le \left(\frac{\alpha_n^{(\ell)} - \beta_n^{(\ell)}}{\alpha_n^{(\ell)} + \beta_n^{(\ell)}}\right)^i \left\|\lambda_n^\ell(0) - \lambda_n^{\ell*}\right\|. \quad (45)$$

For the resources of the ℓth layer, y_s^ℓ, when the upper layer parameters and λ_n^ℓ are fixed and assuming that $\lambda_s^{(\ell-1)*}(y_s^\ell)$ is reached for any y_s^ℓ at the lower layer, the projected gradient method converges to the optimal solution $y_s^{\ell*}$ with a linear rate, i.e., for $s = 1, \ldots, N_\ell$:

$$\left\|y_s^\ell(j) - y_s^{\ell*}\right\| \le \left(\frac{\alpha_s^{(\ell-1)} - \beta_s^{(\ell-1)}}{\alpha_s^{(\ell-1)} + \beta_s^{(\ell-1)}}\right)^j \left\|y_s^\ell(0) - y_s^{\ell*}\right\|. \quad (46)$$

IV. Dynamic Allocation Optimality Bounds

The results of the previous section show that the individual iterations at a given layer converge to the optimal points for a large enough number of iterations, and the optimal results are forwarded to upper layers for them to use in their iterations as gradients. In a practical system with a finite (possibly small) number of iterations, these gradients may have errors and, hence the ideal case of linear convergence no longer applies.

As shown in (28), increasing k and k' causes a decrease in the latency gain. However, they need to be sufficiently large in order to use the convergence results characterized in Thm. 2 in practical scenarios. Hence, there is a trade-off between speed and accuracy. To this end, there is a need for an analysis on the impact of the selection of k and k' on the error. In this section, we first derive some fundamental results on how the gradient error propagates in one layer and then combine these results for our multi-layer approach.

A. Projected Gradient Method With Gradient Errors

Assume that we have a projected gradient descent algorithm with $x(t+1) = \Pi_X\left[x(t) - \alpha g\big(x(t)\big)\right]$ where $\Pi_X[\cdot]$ denotes the projection onto a feasible set and α is a constant step size. The used (erroneous) gradient value $g\big(x(t)\big) = g^*\big(x(t)\big) - E(t)$, where $g^*\big(x(t)\big)$ is the correct gradient value (when the lower layers attain optimum points for a given $x(t)$ in the layered architecture), and $E(t)$ is the error.

Lemma 1: Let the objective function be m-strongly convex and L-smooth. Assuming $0 < \alpha < \frac{2}{L}$, the following holds.

a) $\|x(t+1) - x^*\| \le \Upsilon \|x(t) - x^*\| + \alpha \|E(t)\|$, where $0 < \Upsilon = \max\{|1 - \alpha m|, |1 - \alpha L|\} < 1$.

b) If $\|E(t)\| \le \epsilon$ for all $t > 0$, then

$$\|x(t) - x^*\| \le \Upsilon^t \|x(0) - x^*\| + \frac{1 - \Upsilon^t}{1 - \Upsilon}\alpha\epsilon.$$

Proof: **a)** Since $0 < m < L$, if $0 < \alpha < \frac{2}{L}$, then $0 < \Upsilon < 1$. We characterize the distance from the optimal point x^* as

$$\begin{aligned}
&\|x(t+1) - x^*\| \\
&= \left\|\Pi_X\left[x(t) - \alpha g\big(x(t)\big)\right] - x^* - \alpha g^*\big(x^*\big)\right\| \\
&\le \left\|x(t) - \alpha g\big(x(t)\big) - x^* + \alpha g^*\big(x^*\big)\right\| \\
&= \left\|x(t) - x^* - \alpha\big(g^*(x(t)) - g^*(x^*)\big) + \alpha E(t)\right\| \\
&\le \left\|x(t) - x^* - \alpha\big(g^*(x(t)) - g^*(x^*)\big)\right\| + \alpha\|E(t)\| \\
&\le \max\{|1 - \alpha m|, |1 - \alpha L|\}\|x(t) - x^*\| + \alpha\|E(t)\|,
\end{aligned} \quad (47)$$

where we use the non-expansiveness of the projection and the triangle inequality.

b) Let, $d_i = \|x(i) - x^*\|$. By using Lemma 1.a, we can write

$$\begin{aligned}
d_1 &\le \Upsilon d_0 + \alpha\epsilon, \\
d_2 &\le \Upsilon d_1 + \alpha\epsilon \le \Upsilon^2 d_0 + (\Upsilon + 1)\alpha\epsilon, \\
d_3 &\le \Upsilon d_2 + \alpha\epsilon \le \Upsilon^3 d_0 + (\Upsilon^2 + \Upsilon + 1)\alpha\epsilon, \\
&\vdots \\
d_t &\le \Upsilon^t d_0 + \sum_{i=0}^{t-1}\Upsilon^i\alpha\epsilon = \Upsilon^t d_0 + \frac{1 - \Upsilon^t}{1 - \Upsilon}\alpha\epsilon.
\end{aligned} \quad (48)$$

■

Having characterized the bound on the distance from the optimal point x^* after t iterations and for a given step size α, the following lemma provides the choice of α such that the error bound in Lemma 1.b is smallest.

Lemma 2: Let $B(\alpha) = \Upsilon^t \|x(0) - x^*\| + \frac{1-\Upsilon^t}{1-\Upsilon}\alpha\epsilon$, denote the bound on the error in Lemma 1 after t iterations, where Υ (defined in Lemma 1) also depends on the step size α, where $0 < \alpha < 2/L$.

If $\|x(0) - x^*\| \ge \frac{\epsilon}{m}$, then for any $t > 0$:

$$\alpha^* = \arg\min_\alpha B(\alpha) = \frac{2}{L + m} \quad (49)$$

and

$$B(\alpha^*) = \frac{\epsilon}{m} + \left(\|x(0) - x^*\| - \frac{\epsilon}{m}\right)\left(\frac{L - m}{L + m}\right)^t. \quad (50)$$

If, instead, $\|x(0)-x^*\| < \frac{\epsilon}{m}$, the error bound does not diminish, i.e., $B(\alpha) \geq \|x(0)-x^*\|$.

Proof: Let, $d_0 = \|x(0)-x^*\|$. When $0 \leq \alpha \leq \frac{2}{L+m}$, we have $\Upsilon = 1-\alpha m \leq 1$. Hence, we can write

$$\begin{aligned} B(\alpha) &= \Upsilon^t \|x(0)-x^*\| + \frac{1-\Upsilon^t}{1-\Upsilon}\alpha\epsilon \\ &= \frac{\epsilon}{m} + (d_0 - \frac{\epsilon}{m})(1-\alpha m)^t. \end{aligned} \tag{51}$$

If $d_0 \geq \frac{\epsilon}{m}$, to minimize (51) we need to choose the largest α, which is $\frac{2}{L+m}$. If $d_0 < \frac{\epsilon}{m}$, we need to choose the smallest α, which is zero, where $B(0) = d_0$. This option essentially means keeping the iteration values constant, where the distance from the optimal point stays the same. In that case, choosing a positive step size increases the bound value.

When $\frac{2}{L+m} \leq \alpha \leq \frac{2}{L}$, we have $\Upsilon = \alpha L - 1 \leq 1$. Therefore, we can write

$$\begin{aligned} B(\alpha) &= \Upsilon^t \|x(0)-x^*\| + \frac{1-\Upsilon^t}{1-\Upsilon}\alpha\epsilon \\ &= \frac{\alpha\epsilon}{2-\alpha L} + (d_0 - \frac{\alpha\epsilon}{2-\alpha L})(\alpha L - 1)^t. \end{aligned} \tag{52}$$

If $d_0 - \frac{\alpha\epsilon}{2-\alpha L} \geq 0$, we have $\alpha \leq \frac{2}{L+\frac{\epsilon}{d_0}}$. Since $\alpha \geq \frac{2}{L+m}$, we need to have $\frac{2}{L+\frac{\epsilon}{d_0}} \geq \frac{2}{L+m}$, which essentially means $m \geq \frac{\epsilon}{d_0}$, i.e., $d_0 \geq \frac{\epsilon}{m}$. Therefore, the second term can be nonnegative only if $d_0 \geq \frac{\epsilon}{m}$. The function $B(\alpha)$ is an increasing function of α if $d_0 \geq \frac{\alpha\epsilon}{2-\alpha L}$ and hence, to minimize $B(\alpha)$, we need to choose the smallest α, which is $\frac{2}{L+m}$. If $d_0 < \frac{\epsilon}{m}$, the second term is negative, i.e., $d_0 < \frac{\alpha\epsilon}{2-\alpha L}$, and $B(\alpha) \geq d_0$ when $\frac{2}{L+\frac{\epsilon}{d_0}} < \alpha < \frac{2}{L}$.

Hence, $\alpha^* = \frac{2}{L+m}$ if $d_0 \geq \frac{\epsilon}{m}$. Otherwise, the error bound does not diminish, i.e., $B(\alpha) \geq d_0 = \|x(0)-x^*\|$. ∎

Before getting into the dynamic optimization, we conclude with an observation following directly Lemma 2 (more specifically, from (50)):

Corollary 2: For the projected erroneous gradient method with constant step size $\alpha = \frac{2}{L+m}$, the following holds:

$$\limsup_{t\to\infty} \|x(t)-x^*\| \leq \frac{\epsilon}{m}. \tag{53}$$

This corollary states that the algorithm asymptotically converges to x^* within an error bound ϵ/m with the suitable step size selection.

B. Error Bound of Optimality on H-Stable Optimal Points

We also consider the case where utilities change periodically over time. Specifically, in our model, time is divided into slots, and we assume that the utilities are stationary during the time slot but change from one to the next; hence, the optimal value that the algorithm tries to converge to is constant within a time slot. We define the H-stability of the optimal points in a dynamic setting,[6] as the condition that the difference between the optimal points in two consecutive time slots is upper bounded by the constant H. Let τ denote the time slot index. We assume that $\|x^*(\tau+1) - x^*(\tau)\| \leq H$, where $x^*(\tau)$ is the optimal point for an optimization problem in time slot τ. Assume that we have an algorithm as in Section IV-A with iterations for variable x with a linear rate Υ and gradients with error bound ϵ, i.e., $\|x(t)-x^*\| \leq \Upsilon^t \|x(0)-x^*\| + \frac{1-\Upsilon^t}{1-\Upsilon}\alpha\epsilon$, where $0 < \alpha < 2/L$. Now, assume that we have k iterations in each time slot τ, and we choose the initial point of the iterations in each time slot as the last point that the algorithm has reached in the preceding slot. This heuristic is quite reasonable in dynamic systems that have a limited change of optimal points. The time index of the iterations is specified with t, and $\tau = \lceil \frac{t}{k} \rceil$, where $\lceil \cdot \rceil$ denotes the ceiling function. For this algorithm, we have:

Lemma 3: For the setup described above, the asymptotic error is bounded, i.e.,

$$\limsup_{\tau\to\infty} \|x(\tau k) - x^*(\tau)\| \leq H\frac{\Upsilon^k}{1-\Upsilon^k} + \frac{\alpha\epsilon}{1-\Upsilon}, \tag{54}$$

where this bound can be minimized with constant step size $\alpha = \frac{2}{L+m}$.

Proof: We assume that we choose a random initial feasible point $x(0)$ in the first time slot, and we denote $\xi_0 = x(0) - x^*(1)$. For the end of the computations in the first slot, from Lemma 1.b, we can write

$$\|\xi_1\| = \|x(k) - x^*(1)\| \leq \Upsilon^k \|\xi_0\| + \frac{1-\Upsilon^k}{1-\Upsilon}\alpha\epsilon. \tag{55}$$

Then, at the end of the second slot, we have

$$\|\xi_2\| = \|x(2k) - x^*(2)\| \leq \Upsilon^k \|x(k) - x^*(2)\| + \frac{1-\Upsilon^k}{1-\Upsilon}\alpha\epsilon, \tag{56}$$

since the algorithm starts in time slot 2 with the iterate obtained at the end of the time slot 1. We have

$$\begin{aligned} \|x(k) - x^*(2)\| &= \|x(k) - x^*(1) + x^*(1) - x^*(2)\| \\ &\leq \|x(k) - x^*(1)\| + \|x^*(1) - x^*(2)\| \\ &= \|\xi_1\| + \|x^*(1) - x^*(2)\| \\ &\leq \|\xi_1\| + H. \end{aligned} \tag{57}$$

Therefore, from (55), (56), and (57), we obtain

$$\begin{aligned} \|\xi_2\| &\leq \Upsilon^k(\|\xi_1\| + H) + \frac{1-\Upsilon^k}{1-\Upsilon}\alpha\epsilon \\ &\leq \Upsilon^{2k}\|\xi_0\| + \Upsilon^k H + (\Upsilon^k + 1)\frac{1-\Upsilon^k}{1-\Upsilon}\alpha\epsilon. \end{aligned} \tag{58}$$

Similarly, by using (58), we can write

$$\begin{aligned} \|\xi_3\| &\leq \Upsilon^k(\|\xi_2\| + H) + \frac{1-\Upsilon^k}{1-\Upsilon}\alpha\epsilon \\ &\leq \Upsilon^{3k}\|\xi_0\| + (\Upsilon^{2k}+\Upsilon^k)H + (\Upsilon^{2k}+\Upsilon^k+1)\frac{1-\Upsilon^k}{1-\Upsilon}\alpha\epsilon. \end{aligned}$$

If we continue iteratively, at the end of τ-th slot:

$$\begin{aligned} \|\xi_\tau\| &\leq \Upsilon^{\tau k}\|\xi_0\| + \sum_{j=1}^{\tau-1}\Upsilon^{jk}H + \sum_{i=0}^{\tau-1}\Upsilon^{ik}\frac{1-\Upsilon^k}{1-\Upsilon}\alpha\epsilon \\ &= \Upsilon^{\tau k}\|\xi_0\| + H\frac{\Upsilon^k - \Upsilon^{\tau k}}{1-\Upsilon^k} + \frac{1-\Upsilon^{\tau k}}{1-\Upsilon}\alpha\epsilon. \end{aligned} \tag{59}$$

Thus, when $\tau \to \infty$, since $0 < \Upsilon < 1$, we obtain (54).

We then characterize the optimal step size α minimizing this error bound. Let $\bar{B}(\alpha) = H\frac{\Upsilon^k}{1-\Upsilon^k} + \frac{\alpha\epsilon}{1-\Upsilon}$. We can rewrite this expression as

$$\bar{B}(\alpha) = -H + \frac{H}{1-\Upsilon^k} + \frac{\alpha\epsilon}{1-\Upsilon}. \tag{60}$$

[6]See [26] for a broad analysis of this stability model including estimation of bound H.

When $0 \leq \alpha \leq \frac{2}{L+m}$, we have $\Upsilon = 1-\alpha m \leq 1$, and

$$\bar{B}(\alpha) = -H + \frac{H}{1-(1-\alpha m)^k} + \frac{\epsilon}{m}, \tag{61}$$

which is minimized with the largest α, i.e., $\alpha = \frac{2}{L+m}$.

When $\frac{2}{L+m} \leq \alpha \leq \frac{2}{L}$, we have $\Upsilon = \alpha L - 1 \leq 1$, and

$$\bar{B}(\alpha) = -H + \frac{H}{1-(\alpha L-1)^k} + \frac{\alpha\epsilon}{2-\alpha L}, \tag{62}$$

which is minimized with the smallest α, i.e., $\alpha = \frac{2}{L+m}$, since both the second term and the last term are increasing functions of α. ∎

Lemma 3 shows that the optimality gap is a function of the stability bound H, the gradients errors ϵ, the convergence rate of the method along with the step size, and the number k of iterations in each slot. With this equation, one can decide on the number of necessary iterations to obtain a sufficiently small error level for the H-stable system. In this bound (in (54)), the second term (which comes from errors on gradients) does not disappear with an increasing number of iterations in each slot as expected from Lemma 1.b. These errors originate from the lower layers' iterations in the ML-NUM. Therefore, we can state that these errors cannot be mitigated in the upper layers, and the error characteristics should be evaluated for each layer.

C. Error Bounds for 3-Layer Decomposition Example

Next, we examine the error bounds just obtained for the 3-layer setup in Section III-B, which is critical for designing the multi-layer algorithm appropriately. In Sec. III-B, we defined $(\alpha_o^{(1)}, \beta_o^{(1)})$ and $(\alpha^{(2)}, \beta^{(2)})$ with the equations in (31). Let us introduce the convergence rates as $\Gamma = \max\{|1-\gamma\beta^{(2)}|, |1-\gamma\alpha^{(2)}|\}$ for the iterations in (27), $\Gamma_o' = \max\{|1-\frac{\gamma_o'}{\alpha_o^{(1)}}|, |1-\frac{\gamma_o'}{\beta_o^{(1)}}|\}$ for the iterations in (26), and $\Gamma_o'' = \max\{|1-\gamma_o''\beta_o^{(1)}|, |1-\gamma_o''\alpha_o^{(1)}|\}$ for the iterations in (25). We consider a dynamic (time-varying) NUM and we assume that the optimal global market price is H-stable, i.e., $\|\lambda^*(\tau+1)-\lambda^*(\tau)\| \leq H$, where τ is the time slot index. Then, from (54), for the top layer we write the error bound as

$$\limsup_{\tau\to\infty} \|\lambda(\tau k_\lambda) - \lambda^*(\tau)\| \leq H\frac{\Gamma^{k_\lambda}}{1-\Gamma^{k_\lambda}} + \frac{\gamma\sum_{o=1}^{O}\epsilon_y^{(o)}}{1-\Gamma}, \tag{63}$$

where k_λ is the number of iterations between the τ^{th} and $\tau+1^{\text{st}}$ time slots, and $\epsilon_y^{(o)}$ is the error bound of iterations of y_o for operator $o = 1, \ldots, O$, where their summation is the upper-bound of the gradient errors (corresponds to ϵ in Sec IV-A) in the top layer iterations in (27).

The iterations in (26) for $y_o(i)$ try to converge to the same value $y_o^*(\lambda(j))$ for the amount of time difference between the j^{th} and $j+1^{\text{th}}$ updates of the global price λ. Assume that the change in the $y_o^*(\lambda(j))$ is bounded as $\|y_o^*(\lambda(j+1)) - y_o^*(\lambda(j))\| \leq H_y^{(o)}$. Therefore, for $o = 1, \ldots, O$, the error bound becomes

$$\limsup_{j\to\infty} \|y_o(jk') - y_o^*(j)\| \leq \epsilon_y^{(o)}, \tag{64}$$

$$\epsilon_y^{(o)} \triangleq H_y^{(o)}\frac{(\Gamma_o')^{k'}}{1-(\Gamma_o')^{k'}} + \frac{\gamma_o'\epsilon_\mu^{(o)}}{1-\Gamma_o'}, \tag{65}$$

where k' is the number of iterations between the j^{th} and $j+1^{\text{th}}$ updates of λ, and $\epsilon_\mu^{(o)}$ is the sub-optimality bound of iterations of μ_o which becomes an upper-bound on the gradient errors in (26) for operator $o = 1, \ldots, O$.

Similarly, we extend this analysis to the μ_o updates in (25), where we assume $\|\mu_o^*(y_o(i+1)) - \mu_o^*(y_o(i))\| \leq H_\mu^{(o)}$. Noting that there is no gradient error coming from lower layer iterations, since the end-users calculate the best rates in one shot, we can characterize the error bound of μ_o iterations for $o = 1, \ldots, O$ as

$$\limsup_{i\to\infty} \|\mu_o(ik) - \mu_o^*(i)\| \leq \epsilon_\mu^{(o)}, \tag{66}$$

$$\epsilon_\mu^{(o)} \triangleq H_\mu^{(o)}\frac{(\Gamma_o'')^k}{1-(\Gamma_o'')^k}, \tag{67}$$

where k is the number of iterations between the i^{th} and $i+1^{\text{th}}$ updates of y_o. Therefore, with these bounds, one can characterize the errors in different layers by using the number of iterations for different layers (k_λ, k', k).

Furthermore, from (86) and (27), we have

$$\begin{aligned}\|y_o^*(\lambda(j+1)) - y_o^*(\lambda(j))\| &\leq \alpha_o^{(1)}\|\lambda(j+1)-\lambda(j)\| \\ &\leq \alpha_o^{(1)}\gamma\big(\sum_{o=1}^{O} y_o(jk') - Z\big),\end{aligned} \tag{68}$$

which becomes an upper-bound on $H_y^{(o)}$. Moreover, from (81) and (26),

$$\begin{aligned}\|\mu_o^*(y_o(i+1)) - \mu_o^*(y_o(i))\| &\leq \frac{1}{\beta_o^{(1)}}\|y_o(i+1) - y_o(i)\| \\ &\leq \frac{\gamma_o'}{\beta_o^{(1)}}\big(\mu_o(ik) - \lambda(\lfloor i/k' \rfloor)\big),\end{aligned} \tag{69}$$

which becomes an upper-bound on $H_\mu^{(o)}$. These upper bounds can be used as the estimates of the stability bounds. Therefore, every term in the bounds (63), (64), and (66) is multiplied with the step sizes except the first term with H in (63). Then, we can argue that, in the deterministic case, i.e., $H \approx 0$, the error bounds (63), (64), and (66) get close to zero with sufficiently small step sizes and the algorithm asymptotically converges for any k, k'. More rigorous analysis including the characterization of the convergence rate is left as an open problem for future work.

D. Extension to L-Layer Decomposition

We extend the similar approach to the L-layer case, and we characterize the error bounds by using Lemma 3 as follows.

For the dual variables λ_n^ℓ associated with the constraints of the ℓth layers' resources $\sum_{s\in\mathcal{S}_\ell^n} y_s^\ell \leq y_n^{\ell+1}$ for $n = 1, \ldots, N_{\ell+1}$, the iterations in (41) move towards the same optimal value $\lambda_n^{\ell*}(y_n^{\ell+1}(j))$ between the j^{th} and $j+1^{\text{th}}$ updates of $y_n^{\ell+1}$. Assume $\big\|\lambda_n^{\ell*}\big(y_n^{\ell+1}(j+1)\big) - \lambda_n^{\ell*}\big(y_n^{\ell+1}(j)\big)\big\| \leq H_{\boldsymbol{\lambda}_\ell}^{(n)}$. Then, for $\ell = 2, 3, \ldots, L-1$, we can write the error bound as

$$\begin{aligned}&\limsup_{j\to\infty} \big\|\lambda_n^\ell(jk_\ell) - \lambda_n^{\ell*}(j)\big\| \leq \epsilon_n^\ell, \\ &\epsilon_n^\ell \triangleq H_{\boldsymbol{\lambda}_\ell}^{(n)}\frac{(\Gamma_{\boldsymbol{\lambda}_\ell}^{(n)})^{k_\ell}}{1-(\Gamma_{\boldsymbol{\lambda}_\ell}^{(n)})^{k_\ell}} + \frac{[\mathbf{G}_\ell]_{n,n}\sum_{s\in\mathcal{S}_\ell^n}\varepsilon_s^\ell}{1-\Gamma_{\boldsymbol{\lambda}_\ell}^{(n)}},\end{aligned} \tag{70}$$

where $\Gamma_{\boldsymbol{\lambda}_\ell}^{(n)} = \max\{|1-[\mathbf{G}_\ell]_{n,n}\beta_n^{(\ell)}|, |1-[\mathbf{G}_\ell]_{n,n}\alpha_n^{(\ell)}|\}$, and $[\mathbf{G}_\ell]_{n,n}$ is the n-th diagonal element of step size matrix $[\mathbf{G}_\ell]$ defined in Sec. III-C.

Here, ε_s^ℓ is the sub-optimality bound of the iterations for the resources y_s^ℓ of the ℓ-th layer, and ϵ_n^ℓ (specified in (70)) is the sub-optimality bound of the iterations for the dual variables λ_n^ℓ, where $s \in \mathcal{S}_\ell^n$. The iterations in (42) move towards the same optimal value $y_s^{\ell *}(\lambda_n^\ell(i))$ between the i^{th} and $i+1^{\text{th}}$ updates of λ_n^ℓ. Assume $\left\| y_s^{\ell *}\big(\lambda_n^\ell(i+1)\big) - y_s^{\ell *}\big(\lambda_n^\ell(i)\big) \right\| \leq H_{\boldsymbol{y}_\ell}^{(s)}$. Then, for $\ell = 2, 3, \ldots, L-1$, we can write the error bound as

$$\limsup_{i \to \infty} \left\| \lambda_n^\ell(i k'_\ell) - \lambda_n^{\ell *}(i) \right\| \leq \varepsilon_s^\ell,$$

$$\varepsilon_s^\ell \triangleq H_{\boldsymbol{y}_\ell}^{(s)} \frac{(\Gamma_{\boldsymbol{y}_\ell}^{(s)})^{k'_\ell}}{1-(\Gamma_{\boldsymbol{y}_\ell}^{(s)})^{k'_\ell}} + \frac{[\mathbf{G}'_\ell]_{s,s}\epsilon_s^{\ell-1}}{1-\Gamma_{\boldsymbol{y}_\ell}^{(s)}}, \tag{71}$$

where $\Gamma_{\boldsymbol{y}_\ell}^{(s)} = \max\{|1 - \frac{[\mathbf{G}'_\ell]_{s,s}}{\alpha_s^{(\ell-1)}}|, |1 - \frac{[\mathbf{G}'_\ell]_{s,s}}{\beta_s^{(\ell-1)}}|\}$, and $[\mathbf{G}'_\ell]_{s,s}$ is the s-th diagonal element of step size matrix $[\mathbf{G}'_\ell]$.

V. Numerical Examples

We illustrate our findings in the preceding sections with numerical examples, whereby we mainly focus on comparisons between a 2-Layer algorithm and a 3-Layer algorithm in terms of the performance in dynamic resource allocation scenarios. Our test setup is inspired by distributed weighted waterfilling solutions in radio resource allocation problems [27]. Waterfilling solutions are used in many different problems in communication networks (see [28]–[32] for further details and different applications), we choose a setup from power allocation with independent sub-channels.

We consider $N = 40$ end-users with their utility functions of the form $U_s(x_s) = w_s \log(1 + a_s x_s)$, where a_s is the channel state and w_s is the weight of user s. The total utility is the weighted sum-capacity of Gaussian channels, and the aim of the optimization is maximizing this total utility under a total power constraint:

$$\max_{x_s \in I_s} \sum_{s=1}^{N} w_s \log(1 + a_s x_s) \quad \text{s.t.} \sum_{s=1}^{N} x_s \leq P.$$

We assume the users' weights and channel states are private information, i.e., they are not shared with the others or controllers. Therefore, the central controller does not know the utility functions explicitly and distributed solutions are necessary to solve this allocation problem. We run two different implementations: 1) the 2-Layer algorithm in Eqns. (8)–(9) where the sources directly communicate with the central distributor, and 2) the 3-Layer algorithm in Eqns. (24)–(27) where we have $O = 4$ operators each connected to 10 end-users, i.e., $|\mathcal{S}^o| = 10, o = 1, 2, 3, 4$. We assume that the main source of delay is the communication link with the central distributor also in the 3-Layer algorithm, i.e., the operators are located relatively close to the edge nodes. We model the variations in the channel state a_s according to Rayleigh block fading, and assume the user weights w_s are drawn from a uniform distribution between $[1, 2]$ in each block. Therefore, the optimum allocation we seek changes in every block, and we evaluate the performance in this dynamic allocation scenario (which is equivalent to the H-stable model in Sec. IV-B.). The termination threshold is selected as 10^{-4} at the top layer, where we set $P = 20$ and $I_s = [0, P]$ for all end-users s as total resource and feasibility boundaries, respectively. The step sizes are $\gamma = 1/160$, and $\gamma'_o = 2$, $\gamma''_o = 1/80$ for all operators $o = 1, 2, 3, 4$. We also assume iterations in 2-layer algorithms can be repeated every 100 ms.

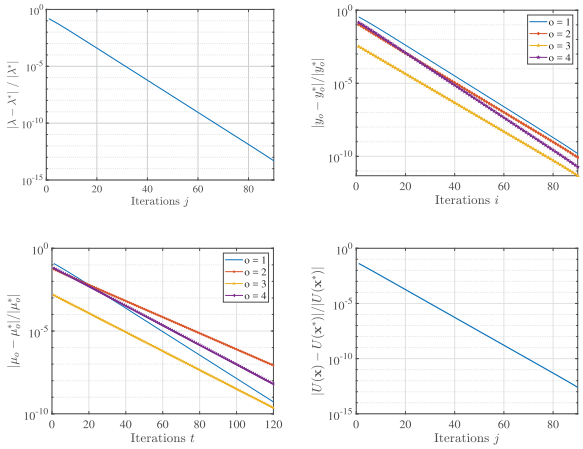
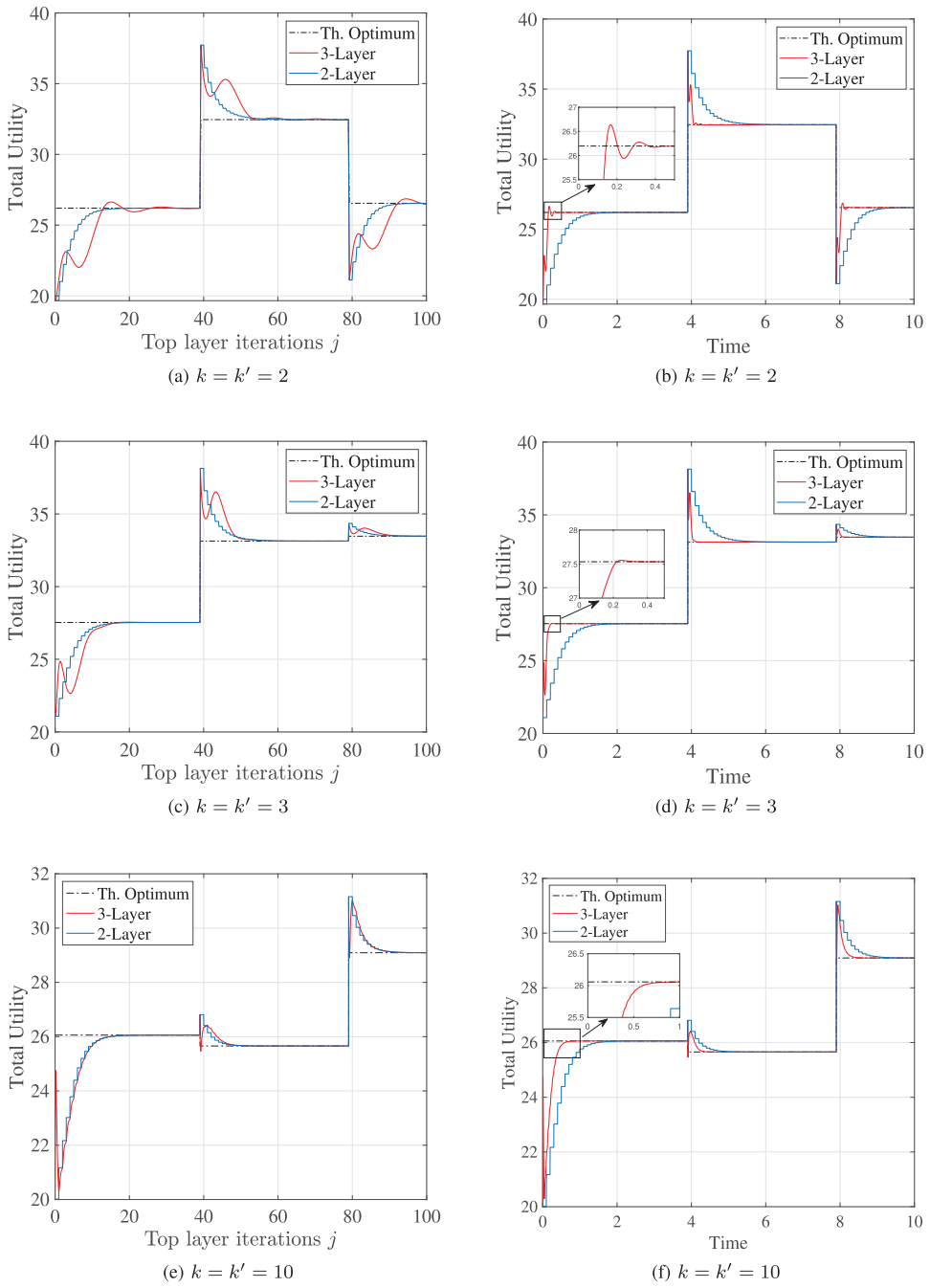


Fig. 4. Linear convergence of individual iterations of λ, y_o, μ_o, and primal function (total utility) $U(\mathbf{x})$.

Before getting into the details of the experiments conducted in the dynamic case, we present the convergence rates of the iterations of different layers along with the primal function value in Fig. 4 for the deterministic case with $k = k' = 200$. As expected, we observe different linear rates for the different operators due to the random nature of the utility functions.

A. Experiment 1: The Effects of Update Frequency Ratios k, k' of the Lower Layer Iterations on Convergence Speed

In a dynamic scenario where channel variations and user weights change every 4 seconds, i.e., in a block fading setup, we compare the convergence behavior of both algorithms with respect to time and with respect to the top layer iteration count for three different update frequency ratios k, k' of lower layer iterations in Fig. 5. In the figure, the convergence behavior for the case when the top layer iteration times are aligned is presented on the left hand side, whereas convergence with respect to time is presented on the right hand side. The latency reduction between the two algorithms is characterized in (28), where $\frac{\rho'}{\rho}$ is set to $1/80$ in our experiment. Ideally, when we have infinitely many iterations in lower layers, the convergence steps should be exactly the same in the top layer for both algorithms due to the equivalence of the problems and the algorithms with the same step size in the top layer. As shown in the left hand side figures (i.e., Fig. 5 (a), (c), (e)), when the numbers k, k' of lower layer iterations decrease, the perturbations from the ideal trajectory increase, i.e., we observe oscillations and lose accuracy to hit the optimum point directly. On the other hand, the latency reduction increases with decreasing numbers k, k' of lower layer iterations as shown in (28). Therefore, there is a trade-off in the selection of the numbers k, k' of lower layer iterations. Lowering them is helpful in terms of latency, however, it also creates perturbations from the ideal trajectory (or ideal linear convergence rate). This implies that one should carefully tune the number of iterations in the lower layers based on the application setup. In our example, as shown in the right hand side figures, reducing the number of iterations from $k, k' = 10$ to 3 in the lower layers reduces the convergence time significantly in the 3-Layer algorithm, however, further reducing it to $k, k' = 2$ does not help, since the perturbations in this case cause slower convergence in terms of the number of top layer iterations. The oscillatory behavior and the convergence times can be seen from the zoomed regions in the right hand side, where $k = k' = 3$ has the best convergence time in this example.



(a) $k = k' = 2$ (b) $k = k' = 2$

(c) $k = k' = 3$ (d) $k = k' = 3$

(e) $k = k' = 10$ (f) $k = k' = 10$

Fig. 5. Comparison of 2-Layer algorithm and 3-Layer algorithm in a dynamic system with various update frequency ratios k, k' of lower layer iterations.

B. Experiment 2: Feasible Resource Allocations

As shown in Fig. 5, the algorithms approach the optimal result from both above and below. The points higher than the optimum are consequences of the constraint violations, i.e., they are infeasible. If the application is strict about the resource distribution constraints, then either the iteration values should be projected onto the feasible region[7] or the results after the convergence should be used. The projection operation onto the set defined with both the resource distribution constraints and the local constraints may be computationally expensive. In these cases, one should use the results after convergence, which highlights the importance of fast convergence, especially if the variations occur frequently.

In the 2-Layer algorithm, the end-users start using the optimum x_s after the convergence criterion is met, since the current iteration values can be infeasible (due to constraint violations) before convergence. That is, they use the previous optimal allocations that have been produced for the previous realization of w_s's and a_s's before full convergence. Similarly, for the middle layer resources (y_o's) in the 3-Layer algorithm, one should wait for full convergence before using them. However, the feasible y_o values of the previous block are used at the beginning of each block, therefore, if the end-users' allocations and local prices (μ_o's) converge in this so-called local market for fixed y_o, these end-users' allocations can be used without waiting for full convergence. Even though

[7]Due to the non-expansiveness of projection operations, our analytical results in the previous sections still hold in this case.

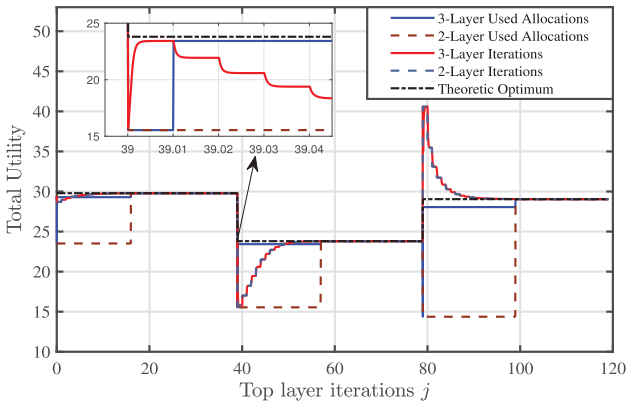
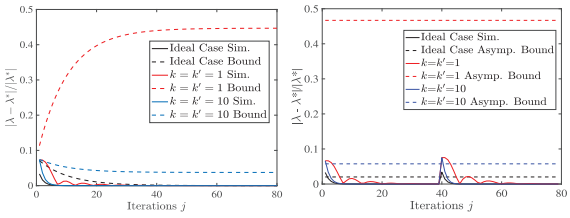


Fig. 6. Comparison of 2-Layer algorithm and 3-Layer algorithm in a dynamic system where only feasible allocations are used. For both the algorithms, used allocations change only when we can guarantee that constraints are satisfied, i.e., iterations converge to a feasible point. The zoom box illustrates the fact that, in a 3-layer algorithm, iterations locally converge to a feasible point earlier due to the convergence of multipliers (μ_o's) in the operators before communicating with the top layer.



(a) Deterministic Case (b) H-Stable Case

Fig. 7. Convergence behaviors of simulations along with the theoretical guarantee bounds.

these allocations are not globally optimal allocations obtained with full convergence, they may provide good sub-optimal allocations. A comparison between 2-Layer algorithm and 3-Layer algorithm in a dynamic setup is given in Fig. 6. In this figure, we present the total utility calculated with both iteration values and used feasible allocations for both algorithms along with the theoretical optimum of the total utility. In order to focus on lower layer iterations, we zoom into a small interval in the same figure, which illustrates that local prices converge to a feasible value. Hence, the results demonstrate that the 3-Layer algorithm reacts immediately, much faster than the 2-Layer algorithm, even though it does not reach the global optimum. We note that in order to obtain this advantageous local convergence, the number k of local price iterations in each update of y_o should be selected to be sufficiently large. We set $k = k' = 100$ in this example, but similar results can be achieved with much smaller values (as shown in the zoomed interval, local prices converge much earlier than the new y_o iterations). Even though we mainly focus on the global convergence of the system in order to provide performance guarantees, the fast reaction due to the local price convergence illustrated here is one of the main significant benefits of the multi-layer algorithm, especially in highly dynamic scenarios.

C. Convergence and Theoretical Error Bounds

In Fig. 7, we present the convergence behavior of the top layer iterations along with the theoretical bounds for the 3-Layer algorithm. We compare the cases where $k = k' = 1$, $k = k' = 10$, and $k = k' = 100$ (which we call the ideal case since lower layers converge before passing any parameters and there are no gradient errors) and we set $a_s = 1, \forall s$.

In Fig. 7a, we visualize the deterministic case in which all the parameters including randomly drawn w_s's stay constant for the entire run, i.e., $H = 0$. For the ideal case, we use the bound in (34) and for the others, we use the bound in Lemma 1, whereby the step sizes specified in Thm. 2 are used. In the ideal case, the bound gets very close to the optimal point after 40 iterations. This means that even in the worst case scenario of randomly drawn w_s's, the algorithm is guaranteed to converge before that. For the other cases, the bounds converge to an upper-bound of the distance from the optimum as discussed in Cor. 2. As shown, the convergence rate is affected by the number of lower layer iterations, where we see a significantly slower convergence with $k = k' = 1$. Also, the upper-bound of the gradient errors ϵ (which is decided empirically) gets quite high while k, k' decreases, and the theoretical bound increases accordingly.

For the dynamic (H-stable) case, where w_s's change randomly every $k_\lambda = 40$ iterations, we present the convergence behavior and the asymptotic performance bounds in Fig. 7b for the same setup. Specifically, we use the asymptotic bound in Lemma 3 (or, correspondingly (63)), where we estimate the parameters H and $\epsilon_y^{(o)}$ empirically after a long-run. As expected, the asymptotic bound (63) gets smaller while k, k' increases due to the decrease of gradient errors $\epsilon_y^{(o)}$. However, it does not reach to zero in the ideal case as in the deterministic case, since the first-term in (63) stays constant with $H > 0$.

VI. Conclusion

We have presented a multi-layer decomposition of the classical NUM problem. The architecture provides a coordination mechanism among multiple operators that are ultimately contending for a shared resource to serve their users. Our convergence analysis and numerical results show that the ML-NUM framework responds faster to changes in a dynamic environment, compared to the standard NUM. In fact, we were able to show that it has the same global convergence rate as the standard NUM framework with two layers, while having a lower communication latency. From the technical analysis stand point, our main contribution is the characterization of the error bound relative to the actual optimum allocation for finite iterations at each layer as well as the proof that there exists an optimum step-size, irrespective of the number of iterations, that minimizes the error bound.

There are several interesting directions for future research on multi-layer decomposition for network utility maximization. This study focused on the maximization of utility as a function of the allocation of a single resource subject to a single constraint, e.g., wireless transmit power. Emerging wireless network virtualization techniques allow for the flexible allocation of a wide range of resources, e.g., buffer capacities, communication channels, and computational resources. Thus, future work could examine utility functions based on multiple resources, each with their own constraints. Moreover, several networking technologies, such as flexible RAN function splits, multi-access edge computing, and networking coding allow the trading off of different types of resources, e.g., allocated computational resources vs. required transmission resources. Other applications exist outside the networking domain, such as the management of *transactive energy* systems. Future research will extend and apply the ML-NUM to optimally trade-off these different resources. At the theoretical level, the study of these applications would likely require to relax

the assumptions made to prove our results to a broader class of utilities and constraints. For instance, our results can be extended to an architecture where lower layer nodes are connected to multiple operators rather than one.[8] Another alternative direction is that some fast algorithms used for standard NUM (see [33], [34]) may be applied to the multi-layer framework. Also, it would be interesting to see if it is possible to establish a similar multi-layered hierarchy in a meshed network scenario where the operators replace the coordinator with a peer-to-peer consensus algorithm, removing the need for a third party acting as a network coordinator, as an alternative to fully decentralized single layer resource allocation algorithms (e.g., [35], [36]).

APPENDIX

A. Proof of Corollary 1

For the Lagrangian function in (4), we have the Hessian matrix with respect to $\mathbf{x}$ as

$$\nabla^2_{\mathbf{x}}(-L(\mathbf{x},\lambda)) = \mathrm{diag}(-\ddot{U}_1(x_1),\ldots,-\ddot{U}_N(x_N)) \succeq \frac{1}{\overline{\alpha}} I$$

since $-\ddot{U}_s(x_s) \geq 1/\alpha_s \geq 1/\overline{\alpha} \geq 0$, and $L(\mathbf{x},\lambda)$ is a strongly concave function of $\mathbf{x}$ with $1/\overline{\alpha}$. Therefore,

$$L(\mathbf{x},\lambda) \leq L(\mathbf{x}^*(\lambda),\lambda) - \frac{1}{2\overline{\alpha}} \left\|\mathbf{x}-\mathbf{x}^*(\lambda)\right\|^2, \forall \mathbf{x} \in \mathcal{I}. \quad (72)$$

If we substitute $\mathbf{x}$ with $\mathbf{x}^*(\lambda^*) = [x_1^*(\lambda^*),\ldots,x_N^*(\lambda^*)]^\top$,

$$\begin{aligned}\frac{1}{2\overline{\alpha}} \left\|\mathbf{x}^*(\lambda^*)-\mathbf{x}^*(\lambda)\right\|^2 &\leq L(\mathbf{x}^*(\lambda),\lambda) - L(\mathbf{x}^*(\lambda^*),\lambda) \\ &= g(\lambda) - g(\lambda^*), \end{aligned} \quad (73)$$

since $g(\lambda) = L(\mathbf{x}^*(\lambda),\lambda)$ and, from (4), $L(\lambda,\mathbf{x}^*(\lambda^*)) = L(\lambda^*,\mathbf{x}^*(\lambda^*)) = g(\lambda^*)$ where $\sum_{s=1}^N x_s^*(\lambda^*) = R_{\mathrm{tot}}$. Then, by using (13) and (73), we obtain (14). We also have

$$\begin{aligned}\left\|\sum_{s=1}^N x_s^*(\lambda(t)) - R_{\mathrm{tot}}\right\| &\leq \sum_{s=1}^N \left\|x_s^*(\lambda(t)) - x_s^*(\lambda^*)\right\| \\ &= \left\|\mathbf{x}^*(\lambda(t)) - \mathbf{x}^*(\lambda^*)\right\|_1 \\ &\leq \sqrt{N}\left\|\mathbf{x}^*(\lambda(t)) - \mathbf{x}^*(\lambda^*)\right\| \end{aligned} \quad (74)$$

where $\|\cdot\|_1$ denotes the ℓ_1 norm. Using (14) and (74), we obtain (15).

B. Proof of Theorem 2

For the 3-layer case, we can modify (11) as follows:

$$\frac{\partial x_s^*(\mu_o,y_o,\lambda)}{\partial \mu_o} = \begin{cases} \dfrac{1}{\ddot{U}_s(x_s^*(\mu_o,y_o,\lambda))}, & \text{for } \dot{U}_s(m_s) \geq \mu_o \geq \dot{U}_s(M_s), \\ 0, & \text{otherwise.} \end{cases} \quad (75)$$

With fixed λ and y_o, for $o = 1,\ldots,O$, from (25) we have

$$\begin{aligned}\frac{\partial^2 L(\boldsymbol{\theta})}{\partial \mu_o^2} &= -\sum_{s\in\mathcal{S}^o} \frac{\partial x_s^*(\mu_o,y_o,\lambda)}{\partial \mu_o} \\ &= -\sum_{s\in\mathcal{C}} \frac{1}{\ddot{U}_s(x_s^*(\mu_o,y_o,\lambda))}, \end{aligned} \quad (76)$$

where the set $\mathcal{C} = \{s \in \mathcal{S}^o : \dot{U}_s(M_s) \leq \mu_o \leq \dot{U}_s(m_s)\}$ has at least one and at most $|\mathcal{S}^o|$ elements as a result of the projection boundaries in (25).

Thus, from **a.2** and **a.5** we have

$$0 < \beta_o^{(1)} = \min_{s\in\mathcal{S}^o}(\beta_s) \leq \frac{\partial^2 L(\boldsymbol{\theta})}{\partial \mu_o^2} \leq |\mathcal{S}^o| \max_{s\in\mathcal{S}^o}(\alpha_s) = \alpha_o^{(1)} < \infty. \quad (77)$$

Then, we obtain (32) by using the rate of convergence derivation in [37, Ch. 2.3].

Now, we go a step further and for fixed λ, we show the convergence of the iterations of y_o. From (26), we can write

$$\frac{\partial L(\boldsymbol{\theta})}{\partial y_o} = \mu_o^*(y_o,\lambda) - \lambda. \quad (78)$$

When μ_o converges to μ_o^*, we can also write

$$\begin{aligned} y_o &= \sum_{s\in\mathcal{S}^o} x_s^*(\mu_o^*(y_o,\lambda)) \\ &= \sum_{s\in\mathcal{C}_1^o} \dot{U}_s^{-1}(\mu_o^*(y_o,\lambda)) + \sum_{s\in\mathcal{C}_2^o} M_s + \sum_{s\in\mathcal{C}_3^o} m_s, \end{aligned} \quad (79)$$

where $\mathcal{C}_1^o = \{s \in \mathcal{S}^o : \dot{U}_s(M_s) \leq \mu_o^*(\lambda,\mathbf{y}) \leq \dot{U}_s(m_s)\}$, $\mathcal{C}_2^o = \{s \in \mathcal{S}^o : \mu_o^*(\lambda,\mathbf{y}) < \dot{U}_s(M_s)\}$, and $\mathcal{C}_3^o = \{s \in \mathcal{S}^o : \mu_o^*(\lambda,\mathbf{y}) > \dot{U}_s(m_s)\}$.
Therefore,

$$\frac{\partial y_o}{\partial \mu_o^*(y_o,\lambda)} = \sum_{s\in\mathcal{C}_1^o} \frac{1}{\ddot{U}_s\big(x_s^*(\mu_o^*(y_o,\lambda))\big)} \quad (80)$$

and, from (78) we have

$$\frac{\partial^2 L(\boldsymbol{\theta})}{\partial y_o^2} = \frac{\partial \mu_o^*(y_o,\lambda)}{\partial y_o}. \quad (81)$$

Thus,

$$\left|\frac{\partial^2 L(\boldsymbol{\theta})}{\partial y_o^2}\right| = \left(\sum_{s\in\mathcal{C}_1^o} -\frac{1}{\ddot{U}_s\big(x_s^*(\mu_o^*(y_o,\lambda))\big)}\right)^{-1} \quad (82)$$

and since $\mathcal{C}_1^o$ has at least one and at most $|\mathcal{S}^o|$ elements, we have

$$\frac{1}{\alpha_o^{(1)}} = \frac{1}{|\mathcal{S}^o| \max_{s\in\mathcal{S}^o}(\alpha_s)} \leq \left|\frac{\partial^2 L(\boldsymbol{\theta})}{\partial y_o^2}\right| \leq \frac{1}{\min_{s\in\mathcal{S}^o}(\beta_s)} = \frac{1}{\beta_o^{(1)}}. \quad (83)$$

Then, we obtain (33) by using the rate of convergence derivation in [37, Ch. 2.3].

Lastly, we investigate the rate of convergence of the iterations for λ. From (27), we have

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \lambda} = Z - \sum_{o=1}^O y_o^*(\lambda). \quad (84)$$

When y_o has converged, we have $\lambda = \mu_o^*(y_o^*(\lambda))$, and we can write

$$y_o^*(\lambda) = \sum_{s\in\mathcal{C}_1^o} \dot{U}_s^{-1}(\lambda) + \sum_{s\in\mathcal{C}_2^o} M_s + \sum_{s\in\mathcal{C}_3^o} m_s. \quad (85)$$

Then, we can write

$$\frac{\partial y_o^*(\lambda)}{\partial \lambda} = \sum_{s\in\mathcal{C}_1^o} \frac{1}{\ddot{U}_s(x_s^*(\lambda))}, \quad (86)$$

[8] This type of architectures are common in flow networks, where this resource allocation is not an equivalent of the classical problem in (1). Noting that, our results can be extended in a slightly straightforward manner to these scenarios by following similar steps of the proofs.

which leads to

$$\frac{\partial^2 L(\boldsymbol{\theta})}{\partial \lambda^2} = -\sum_{o=1}^{O} \frac{\partial y_o^*(\lambda)}{\partial \lambda} = -\sum_{o=1}^{O} \sum_{s \in \mathcal{C}_1^o} \frac{1}{\ddot{U}_s(x_s^*(\lambda))} \quad (87)$$

and with (30), we have

$$0 < \beta^{(2)} \leq \frac{\partial^2 L(\boldsymbol{\theta})}{\partial \lambda^2} \leq \alpha^{(2)} < \infty. \quad (88)$$

Similarly, we obtain (34) by using the rate of convergence derivation in [37, Ch. 2.3].

C. Extension of the Theorem 2 to the L-Layer Decomposition

For λ_n^ℓ, we can write

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \lambda_n^\ell} = y_n^{\ell+1} - \sum_{a \in \mathcal{S}_\ell^n} y_a^{\ell *}(\lambda_n^\ell). \quad (89)$$

Since $y_a^{\ell *}$ has converged, we have $\lambda_n^\ell = \lambda_a^{\ell-1}$ where $a \in \mathcal{S}_\ell^n$. Thus, we have

$$y_a^{\ell *}(\lambda_n^\ell) = \sum_{s \in \overline{\mathcal{C}}_1^a} \dot{U}_s^{-1}(\lambda_n^\ell) + \sum_{s \in \overline{\mathcal{C}}_2^a} M_s + \sum_{s \in \overline{\mathcal{C}}_3^a} m_s,$$
$$\frac{\partial y_a^{\ell *}}{\partial \lambda_n^\ell} = \sum_{s \in \overline{\mathcal{C}}_1^a} \frac{1}{\ddot{U}_s(y_s^1(\lambda_n^\ell))}. \quad (90)$$

Here, $\overline{\mathcal{C}}_1^a = \{s \in \mathcal{X}_\ell^a : \dot{U}_s(M_s) \leq \lambda_n^\ell \leq \dot{U}_s(m_s)\}$, $\overline{\mathcal{C}}_2^a = \{s \in \mathcal{X}_\ell^a : \lambda_n^\ell < \dot{U}_s(M_s)\}$, and $\overline{\mathcal{C}}_3^a = \{s \in \mathcal{X}_\ell^a : \lambda_n^\ell > \dot{U}_s(m_s)\}$, where $\mathcal{X}_\ell^a$ denotes the set of all layer-1 elements that are part of the subtree of layer-ℓ element $a = 1, \ldots, N_\ell$. Then, we can write

$$\begin{aligned} \frac{\partial^2 L(\boldsymbol{\theta})}{(\partial \lambda_n^\ell)^2} &= -\sum_{a \in \mathcal{S}_\ell^n} \frac{\partial y_a^{\ell *}}{\partial \lambda_n^\ell} \\ &= -\sum_{a \in \mathcal{S}_\ell^n} \sum_{s \in \overline{\mathcal{C}}_1^a} \frac{1}{\ddot{U}_s(y_s^1(\lambda_n^\ell))} \\ &= -\sum_{s \in \overline{\mathcal{C}}_1^n} \frac{1}{\ddot{U}_s(y_s^1(\lambda_n^\ell))}, \end{aligned} \quad (91)$$

where $\overline{\mathcal{C}}_1^n = \{s \in \mathcal{X}_{\ell+1}^n : \dot{U}_s(M_s) \leq \lambda_n^\ell \leq \dot{U}_s(m_s)\}$, and we have

$$0 < \beta_n^{(\ell)} \leq \frac{\partial^2 L(\boldsymbol{\theta})}{(\partial \lambda_n^\ell)^2} \leq \alpha_n^{(\ell)} < \infty. \quad (92)$$

Then, we obtain (45) by using the rate of convergence derivation in [37, Ch. 2.3].

For y_a^ℓ, we have

$$\frac{\partial L(\boldsymbol{\theta})}{\partial y_a^\ell} = \lambda_a^{(\ell-1)*}(y_a^\ell) - \lambda_n^\ell. \quad (93)$$

When $\lambda_a^{\ell-1}$ has converged, we have

$$\begin{aligned} y_a^\ell &= \sum_{s \in \mathcal{X}_\ell^a} \dot{U}_s^{-1}(\lambda_a^{(\ell-1)*}) \\ &= \sum_{s \in \overline{\mathcal{C}}_1^a} \dot{U}_s^{-1}(\lambda_a^{(\ell-1)*}) + \sum_{s \in \overline{\mathcal{C}}_2^a} M_s + \sum_{s \in \overline{\mathcal{C}}_3^a} m_s. \end{aligned} \quad (94)$$

Thus, we have

$$\frac{\partial y_a^\ell}{\partial \lambda_a^{(\ell-1)*}} = \sum_{s \in \overline{\mathcal{C}}_1^a} \frac{1}{\ddot{U}_s\big(y_s^1(\lambda_a^{(\ell-1)*})\big)}, \quad (95)$$

and

$$\frac{\partial^2 L(\boldsymbol{\theta})}{(\partial y_a^\ell)^2} = \frac{\partial \lambda_a^{(\ell-1)*}}{\partial y_a^\ell}. \quad (96)$$

Then, we write

$$\left| \frac{\partial^2 L(\boldsymbol{\theta})}{(\partial y_a^\ell)^2} \right| = \Big(\sum_{s \in \overline{\mathcal{C}}_1^a} \frac{1}{\ddot{U}_s\big(y_s^1(\lambda_a^{(\ell-1)*})\big)} \Big)^{-1}, \quad (97)$$

which leads to

$$0 < \frac{1}{\alpha_a^{(\ell-1)}} \leq \left| \frac{\partial^2 L(\boldsymbol{\theta})}{(\partial y_a^\ell)^2} \right| \leq \frac{1}{\beta_a^{(\ell-1)}} < \infty, \quad (98)$$

where it is enough to show (46) by using the rate of convergence derivation in [37, Ch. 2.3].

REFERENCES

[1] N. Karakoc, A. Scaglione, and A. Nedic, "Multi-layer decomposition of optimal resource sharing problems," in *Proc. IEEE Conf. Decis. Control (CDC)*, Miami Beach, FL, USA, Dec. 2018, pp. 1–6.

[2] L. Ferrari, N. Karakoc, A. Scaglione, M. Reisslein, and A. Thyagaturu, "Layered cooperative resource sharing at a wireless SDN backhaul," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Kansas City, MO, USA, May 2018, pp. 1–6.

[3] P. Shantharama, A. S. Thyagaturu, N. Karakoc, L. Ferrari, M. Reisslein, and A. Scaglione, "LayBack: SDN management of multi-access edge computing (MEC) for network access services and radio resource sharing," *IEEE Access*, vol. 6, pp. 57545–57561, 2018.

[4] M. Wang *et al.*, "A multi-layer multi-timescale network utility maximization framework for the SDN-based LayBack architecture enabling wireless backhaul resource sharing," *Electronics*, vol. 8, no. 9, pp. 937.1–937.28, Aug. 2019.

[5] J.-B. Sheu, "A novel dynamic resource allocation model for demand-responsive city logistics distribution operations," *Transp. Res. E: Logistics Transp. Rev.*, vol. 42, no. 6, pp. 445–472, Nov. 2006.

[6] Z. Zhou, M. Dong, K. Ota, and Z. Chang, "Energy-efficient context-aware matching for resource allocation in ultra-dense small cells," *IEEE Access*, vol. 3, pp. 1849–1860, 2015.

[7] Z. Liu, Q. Wu, S. Huang, and H. Zhao, "Transactive energy: A review of state of the art and implementation," in *Proc. IEEE Manchester PowerTech*, Jun. 2017, pp. 1–6.

[8] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *J. Oper. Res. Soc.*, vol. 49, no. 3, pp. 237–252, Apr. 1998.

[9] S. H. Low and D. E. Lapsely, "Optimization flow control. I. basic algorithm and convergence," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 861–874, Dec. 1999.

[10] M. Chiang, S. H. Low, R. Calderbank, and J. C. Doyle, "Layering as optimization decomposition," *Proc. IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.

[11] B. Johansson, P. Soldati, and M. Johansson, "Mathematical decomposition techniques for distributed cross-layer optimization of data networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1535–1547, Aug. 2006.

[12] D. P. Palomar and M. Chiang, "Alternative distributed algorithms for network utility maximization: Framework and applications," *IEEE Trans. Autom. Control*, vol. 52, no. 12, pp. 2254–2269, Dec. 2007.

[13] X. Lin, N. B. Shroff, and R. Srikant, "On the connection-level stability of congestion-controlled communication networks," *IEEE Trans. Inf. Theory*, vol. 54, no. 5, pp. 2317–2338, May 2008.

[14] D. P. Bertsekas and J. N. Tsitsiklis, "Gradient convergence in gradient methods with errors," *SIAM J. Optim.*, vol. 10, no. 3, pp. 627–642, Jan. 2000.

[15] A. Nedić and D. P. Bertsekas, "The effect of deterministic noise in subgradient methods," *Math. Program.*, vol. 125, no. 1, pp. 75–99, Sep. 2010.

[16] M. Mehyar, D. Spanos, and S. H. Low, "Optimization flow control with estimation error," in *Proc. IEEE INFOCOM*, Mar. 2004, pp. 984–992.

[17] J. Zhang, D. Zheng, and M. Chiang, "The impact of stochastic noisy feedback on distributed network utility maximization," *IEEE Trans. Inf. Theory*, vol. 54, no. 2, pp. 645–665, Feb. 2008.

[18] Y. Cui and V. K. N. Lau, "Convergence-optimal quantizer design of distributed contraction-based iterative algorithms with quantized message passing," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5196–5205, Oct. 2010.
[19] N. Gatsis and G. B. Giannakis, "Power control with imperfect exchanges and applications to spectrum sharing," *IEEE Trans. Signal Process.*, vol. 59, no. 7, pp. 3410–3423, Jul. 2011.
[20] A. Y. Popkov, "Gradient methods for nonstationary unconstrained optimization problems," *Autom. Remote Control*, vol. 66, no. 6, pp. 883–891, Jun. 2005.
[21] C. Xi and U. A. Khan, "Distributed dynamic optimization over directed graphs," in *Proc. IEEE 55th Conf. Decis. Control (CDC)*, Dec. 2016, pp. 245–250.
[22] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Trans. Netw.*, vol. 8, no. 5, pp. 556–567, Oct. 2000.
[23] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, Jan. 2009.
[24] I. Necoara and A. Patrascu, "Iteration complexity analysis of dual first-order methods for conic convex programming," *Optim. Methods Softw.*, vol. 31, no. 3, pp. 645–678, May 2016.
[25] H. Yu and M. J. Neely, "On the convergence time of dual subgradient methods for strongly convex programs," *IEEE Trans. Autom. Control*, vol. 63, no. 4, pp. 1105–1112, Apr. 2018.
[26] C. Wilson, V. V. Veeravalli, and A. Nedic, "Adaptive sequential stochastic optimization," *IEEE Trans. Autom. Control*, vol. 64, no. 2, pp. 496–509, Feb. 2019.
[27] P. He, L. Zhao, S. Zhou, and Z. Niu, "Water-filling: A geometric approach and its application to solve generalized radio resource allocation problems," *IEEE Trans. Wireless Commun.*, vol. 12, no. 7, pp. 3637–3647, Jul. 2013.
[28] D. P. Palomar and J. R. Fonollosa, "Practical algorithms for a family of waterfilling solutions," *IEEE Trans. Signal Process.*, vol. 53, no. 2, pp. 686–695, Feb. 2005.
[29] A. Scaglione, G. B. Giannakis, and S. Barbarossa, "Redundant filterbank precoders and equalizers. I. Unification and optimal designs," *IEEE Trans. Signal Process.*, vol. 47, no. 7, pp. 1988–2006, Jul. 1999, doi: 10.1109/78.771047.
[30] E. Telatar, "Capacity of multi-antenna Gaussian channels," *Eur. Trans. Telecommun.*, vol. 10, no. 6, pp. 585–595, Nov. 1999.
[31] A. Scaglione, S. Barbarossa, and G. B. Giannakis, "Filterbank transceivers optimizing information rate in block transmissions over dispersive channels," *IEEE Trans. Inf. Theory*, vol. 45, no. 3, pp. 1019–1032, Apr. 1999.
[32] A. Scaglione, S. Barbarossa, and G. B. Giannakis, "Optimal adaptive precoding for frequency-selective Nagakami-m fading channels," in *Proc. Veh. Technol. Conf. Fall IEEE VTS Fall VTC. 52nd Veh. Technol. Conf.*, vol. 3, Sep. 2000, pp. 1291–1295.
[33] A. Beck, A. Nedić, A. Ozdaglar, and M. Teboulle, "An $O(1/k)$ gradient method for network resource allocation problems," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 1, pp. 64–73, Mar. 2014.
[34] M. Zargham, A. Ribeiro, A. Ozdaglar, and A. Jadbabaie, "Accelerated dual descent for network flow optimization," *IEEE Trans. Autom. Control*, vol. 59, no. 4, pp. 905–920, Apr. 2014.
[35] Y. Ho, L. Servi, and R. Suri, "A class of center-free resource allocation algorithms," *Large Scale Syst.*, vol. 1, no. 1, pp. 51–62, 1980.
[36] L. Xiao and S. Boyd, "Optimal scaling of a gradient method for distributed resource allocation," *J. Optim. Theory Appl.*, vol. 129, no. 3, pp. 469–488, Dec. 2006.
[37] D. Bertsekas, *Nonlinear Programming*. Belmont, MA, USA: Athena Scientific, 1999.

Nurullah Karakoç (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical and electronics engineering from Bilkent University, Turkey, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree with Arizona State University, Tempe. He was with Google as a Summer Intern in 2019. His current research interests include wireless communications, networking, and optimization theory.

Anna Scaglione (Fellow, IEEE) received the M.Sc. and Ph.D. degrees in 1995 and 1999, respectively. She is currently a Professor in electrical and computer engineering with Arizona State University. Her research is rooted in statistical signal processing and spans many disciplines that relate to network science, including communication, control, and energy-delivery systems. Her most recent work in signal processing focused on distributed learning and data analytics for signals that are driven by network processes. She received the 2000 IEEE TRANSACTIONS ON SIGNAL PROCESSING Best Paper Award and the 2013 IEEE Donald G. Fink Prize Paper Award. Her work with her students earned several conference paper awards in addition to the 2013 IEEE Signal Processing Society Young Author Best Paper Award (Lin Li). She is a Distinguished Lecturer for the IEEE Signal Processing Society from 2019 to 2020. She is the Deputy Editor-in-Chief of the IEEE TRANSACTIONS ON CONTROL OVER NETWORKED SYSTEMS.

Angelia Nedić (Member, IEEE) received the Ph.D. degree in computational mathematics and mathematical physics from Moscow State University, Moscow, Russia, in 1994, and the Ph.D. degree in electrical and computer science engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2002. She has worked as a Senior Engineer with BAE Systems North America, Advanced Information Technology Division, Burlington, MA. She is currently a Faculty Member of the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe. Prior to joining Arizona State University, she has been a Willard Scholar Faculty Member at the University of Illinois at Urbana–Champaign. Her general research interests include optimization, large-scale complex systems dynamics, variational inequalities, and games. She was a recipient (jointly with her coauthors) of the Best Paper Award at the Winter Simulation Conference 2013 and the Best Paper Award at the International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt) 2015.

Martin Reisslein (Fellow, IEEE) received the Ph.D. degree in systems engineering from the University of Pennsylvania in 1998. He is a Professor with the School of Electrical, Computer, and Energy Engineering, Arizona State University (ASU), Tempe. He chaired the Steering Committee of the IEEE TRANSACTIONS ON MULTIMEDIA from 2017 to 2019 and was an Associate Editor of the IEEE/ACM TRANSACTIONS ON NETWORKING from 2009 to 2013. He received the IEEE Communications Society Best Tutorial Paper Award in 2008, the Friedrich Wilhelm Bessel Research Award from the Alexander von Humboldt Foundation in 2015, as well as a DRESDEN Senior Fellowship in 2016 and 2019. He currently serves as an Associate Editor for the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON EDUCATION, IEEE ACCESS, and *Computer Networks*. He is currently an Associate Editor-in-Chief of the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS and the Co-Editor-in-Chief of *Optical Switching and Networking*.