Federated Network Utility Maximization

Nurullah Karakoç, Anna Scaglione
School of Electrical, Computer and Energy Engineering
Arizona State University, Tempe, AZ
nkarakoc, ascaglio@asu.edu

***Abstract*—We consider a large-scale optimization problem with private utilities over a network where servers are connected through a graph and each server has their own edge devices connected to it. This setup finds applications in many network architectures from wireless communications to power networks and to supply chain management. Over this architecture, we provide a decentralized federated optimization algorithm that uses decomposition methods along with a single round of consensus exchange on the global dual variables. The convergence performance of the method is illustrated with numerical examples.**

I. Introduction

With the proliferation of devices connected to the networks, data processing requires more and more distributed computing in the networks that are growing constantly and rapidly in terms of scale. Considering the fact that data is produced in edge devices, shifting computation closer to the edge side increases efficiency while bypassing big data transmissions [1]. Using the cloud for computing has obvious advantages such as much powerful hardware, sharing of resources etc. compared to the edge devices which are powered by batteries mostly. On the other hand, edge computing has significant benefits mostly due to reduced delay and offloading of the communication burden of the cloud network. In addition, it keeps the data in the edge devices which reduces privacy concerns of users significantly. To this end, for example, big companies like Apple and Google who produce consumer electronics try to bring dedicated processing hardware (named Neural Engine and Neural Core) to their products for on-device machine learning applications which is critical for real-time processing. We expect to see the continuation of this trend in the near future in order to support many delay-critical and privacy-sensitive tasks and applications.

The algorithms harnessing the end devices' capabilities require a divide and conquer approach due to the scale. Nowadays, the decomposable problems are quickly mapped into the distributed optimization algorithms [2], where the problem is generally divided into smaller parts assigned to different entities and the coordination is executed through message passing schemes. These problems are generally expressed with sum-utility objectives along with local and global constraints. Many problems from the areas of machine learning/data fitting to the resource allocation and coordination fit well into these formulations where the decomposition of the overall problem into sub-problems is possible.

This work is supported by NSF NeTS: 1716121.

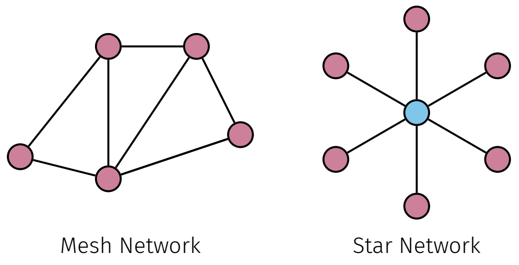

Fig. 1: Mesh vs Star Networks.

In general, there are two different flavors of the considered architectures which we name them as star networks and mesh networks as illustrated in Fig. 1. Communication schemes are generally implemented as master-slave (also called master-worker) in the star and peer-to-peer in the mesh networks. In the master-slave scheme, the slave nodes are responsible for the parallel updates and the master node combines these updates and broadcast it back to the other nodes. On the other hand, in the peer-to-peer, nodes aggregate their info with the other nodes directly through a message exchange scheme (i.e., an aggregation rule) after their parallel computations.

The optimization over the star networks is typically studied under the network utility maximization (NUM) framework, where fair resource allocation to heterogeneous users is formulated as a convex problem [3], [4] under link capacity constraints in communication networks. The solutions under this framework mostly use primal/dual decomposition (see [5], [6] for various decomposition techniques) where in the distributed solution, the slave nodes compute the primal updates whereas the master node is responsible from the dual updates that use aggregated values of the primal variables. In prior work [7]–[9], we have proposed a layered architecture and a multi-layer hierarchical NUM algorithm for resource allocation and coordination of radio access networks using a Software Defined Network (SDN) orchestrator in the backhaul, where the convergence of the algorithm is discussed in [10]. One main consideration in these works is that we assume the changes in the system occur at the same time-scale with the algorithm, where there are fixed number of iterations allowed for different updates as opposed to the commonly used time-scale separation assumption in which the algorithms converge asymptotically before any change and each update is assumed have enough iterations (or time) to converge. Using fixed

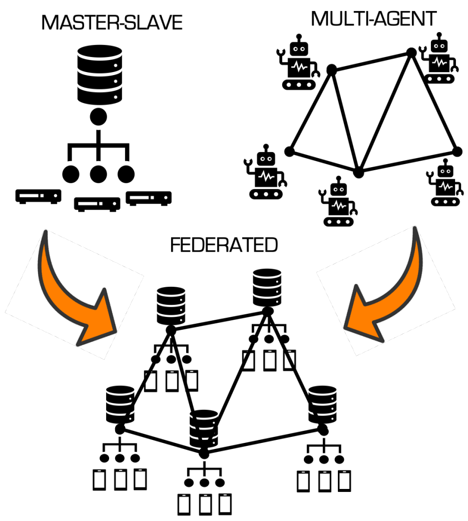

Fig. 2: Considered setup is a mesh network of star networks.

number of iterations brings an important limitation to the algorithm design, where the effects of these iteration counts to the convergence and to the possible errors caused by inexact (incomplete) computations should be evaluated. Another well-studied algorithm over star networks is Alternating Direction Method of Multipliers [11] (ADMM) which provides improved convergence properties with the use of Augmented Lagrangians over dual methods used by NUM.

Distributed optimization methods [12]–[14] over the mesh networks typically use consensus (i.e., distributed averaging) algorithms [15], [16] as message exchange schemes where the aim is mostly to find a global optimizer of a common problem. These algorithms converge asymptotically, and they can be called approximate distributed averaging schemes in finite time implementations. On the other hand, exact averaging (e.g., via AllReduce) approaches are sensitive to stragglers and delays in the communication links [17]. In addition, consensus algorithms are not fully sequential (like AllReduce) which allows parallelization of the computations and relaxes synchronization requirements. Therefore, there are certain advantages in using approximate methods, however, they create additional noise.

In this work, we consider an architecture where servers are connected over a graph, and each server has its own edge devices connected to it. One can think of it as the juxtaposition of a mesh network and star networks as illustrated in Fig. 2. This architecture is a natural fit to many networks from broad range of applications. A typical example is from the wireless networks, where end devices are connected to the access points (e.g., WiFI APs, eNBs etc.) that are connected to each other through a mesh network. Another examples can be found from power networks or market modelling from economics. Over this architecture, we provide an optimization framework and an algorithm to solve the problems with sum-utility objectives along with decomposable local (in-server) and global (network-wide) constraints. The utilities are assumed private and unknown to any other entity than the particular edge node. The developed method is based on the usage of primal-dual methods combined with a single round of consensus exchange on the global dual variables. We essentially bring together the vertical multi-layer decomposition enabled by NUM framework and the horizontal decomposition of consensus algorithms. We see this work as just an introductory effort towards this direction, where many open research directions such as the theoretical analysis of convergence, the error analysis in finite-time implementations, the performance under different consensus schemes, the combination of the algorithm with the ADMM for broader set of objective functions etc. deserve further investigation. There is a vast literature both on the decomposition theory and on the consensus algorithms, and the consideration of them together in architectures can lead to holistic practical designs.

II. Problem Formulation

We consider a network of N servers (agents) connected over an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with set of vertices $\mathcal{V}$ ($|\mathcal{V}| = N$) and set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Each server $n = 1, 2, \ldots, N$ in this network has a set of workers, (i.e., edge devices) denoted with $\mathcal{S}_n$. We assume these edge devices do not have connectivity among themselves and their only connectivity to the network is through their unique servers, i.e., if a worker $i \in \mathcal{S}_n$ then $i \notin \mathcal{S}_{n'}$, $\forall n' \neq n$. Over this architecture, we consider a network-wide total utility maximization (or cost/lost function minimization), where each edge device has their own private utility function which is assumed to be unknown by any other entity in the network. In addition, we consider two sets of coupling constraints: the local constraints that couple the variables of the edge devices connected to the same server and the global constraints that couple the servers' variables network-wide. We assume the local constraints and the global constraints are decomposable over edge devices and servers, respectively. There may also be specific and private constraints that affect the variables exclusively associated with servers or edge devices (such as upper/lower bounds on particular variables). Mathematically, we can formulate the optimization problem as

$$\begin{aligned} \max \sum_{n\in\mathcal{V}} \sum_{i\in\mathcal{S}_n} & f_i(\mathbf{x}_i) \\ \text{s.t.} \sum_{i\in\mathcal{S}_n} & \mathbf{h}_i(\mathbf{x}_i, \boldsymbol{y}_n) \leq \mathbf{0}, \forall n \in \mathcal{V} \text{ (server constraints)} \\ \sum_{n\in\mathcal{V}} & \mathbf{g}_n(\boldsymbol{y}_n) \leq \mathbf{0}, \qquad \text{(global constraints)} \\ & \boldsymbol{y}_n \in \mathcal{Y}_n, \mathbf{x}_i \in \mathcal{X}_i, \forall i, n, \end{aligned} \tag{1}$$

where $\mathbf{x}_i$ denotes the optimization variable of worker i, and $\boldsymbol{y}_n$ denotes the optimization variable of server n. In addition, f_i denotes private utility (or cost for the minimization case) function of edge device i, $\mathbf{h}_i$ and $\mathbf{g}_n$ are local and global constraint functions (possibly vector-valued mappings), respectively and the convex sets $\mathcal{Y}_n, \mathcal{X}_i$ are private constraints[1]. The summation and the inequality operations are understood coordinate-wise.

[1]We denote the vectors and matrices with lowercase bold and uppercase bold letters (e.g., $\mathbf{x}_i$ is a vector and $\mathbf{W}$ is a matrix), respectively, whereas the entries are denoted by subscripts after brackets (e.g., $[\mathbf{x}]_i$ and $[\mathbf{W}]_{ij}$).

The typical examples arise from resource allocation problems, where the local constraints are distribution of the server resources, and the global constraints are the network-wide allocation of resources (e.g., see Sec. IV). Nonetheless, our formulation and the solution procedure apply to the problems that can be formulated/re-formulated in similar form as (1).

We can write the Lagrangian as

$$\begin{aligned} L(\mathbf{x},\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\mu}) = &\sum_{n\in\mathcal{V}}\sum_{i\in\mathcal{S}_n} f_i(\mathbf{x}_i) - \sum_{n\in\mathcal{V}}\boldsymbol{\mu}_n^\top \sum_{i\in\mathcal{S}_n}\mathbf{h}_i(\mathbf{x}_i,\boldsymbol{y}_n) \\ &- \boldsymbol{\lambda}^\top \sum_{n\in\mathcal{V}}\mathbf{g}_n(\boldsymbol{y}_n), \end{aligned} \tag{2}$$

where $\boldsymbol{\mu}_n$ is the multiplier associated with the local constraint of node n, and $\boldsymbol{\lambda}$ is the multiplier associated with the global constraint. Also, $\mathbf{x},\boldsymbol{y}$ and $\boldsymbol{\mu}$ contains all edge devices' $\mathbf{x}_i$'s, all servers' $\boldsymbol{y}_n$'s and $\boldsymbol{\mu}_n$'s, respectively. Therefore, the dual objective becomes

$$D(\boldsymbol{\lambda},\boldsymbol{\mu}) = \max_{\boldsymbol{y}_n\in\mathcal{Y}_n,\mathbf{x}_i\in\mathcal{X}_i,\forall i,n} L(\mathbf{x},\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\mu}), \tag{3}$$

and the dual problem is

$$\min_{\lambda\geq 0,\mu_n\geq 0,\forall n} D(\boldsymbol{\lambda},\boldsymbol{\mu}). \tag{4}$$

Assuming strong duality, a decentralized and distributed solution of the problem should aim to find optimal values of the primal variables $\mathbf{x},\boldsymbol{y}$ and the dual variables $\boldsymbol{\lambda},\boldsymbol{\mu}$ in a center-free and a rapid way. In the next section, we propose a decentralized algorithm that reduces communication and computation burden by shifting them to the edges while responding quickly to changes.

III. Decentralized Solution via Decomposition

As a first step, we rearrange the Lagrangian in (2) as

$$\begin{aligned} L(\mathbf{x},\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\mu}) = &\sum_{n\in\mathcal{V}}\left(\sum_{i\in\mathcal{S}_n} f_i(\mathbf{x}_i) - \sum_{i\in\mathcal{S}_n}\boldsymbol{\mu}_n^\top \mathbf{h}_i(\mathbf{x}_i,\boldsymbol{y}_n)\right) \\ &- \boldsymbol{\lambda}^\top \sum_{n\in\mathcal{V}}\mathbf{g}_n(\boldsymbol{y}_n), \end{aligned} \tag{5}$$

where the part in the parenthesis does not depend on $\boldsymbol{\lambda}$, and therefore, for fixed $\boldsymbol{y}_n$, it can be treated as a sub-problem with primal variable $\mathbf{x}_i$'s and a dual variable $\boldsymbol{\mu}_n$. It is in fact the Lagrangian of the sub-problem in each server n:

$$\begin{aligned} &\max_{\mathbf{x}_i\in\mathcal{X}_i,i\in\mathcal{S}_n} \sum_{i\in\mathcal{S}_n} f_i(\mathbf{x}_i) \\ &\text{s.t.} \quad \sum_{i\in\mathcal{S}_n}\mathbf{h}_i(\mathbf{x}_i,\boldsymbol{y}_n) \leq \mathbf{0}. \end{aligned} \tag{6}$$

For fixed $\boldsymbol{y}_n$, this sub-problem can be solved among each server and its clients. Then the network-wide problem becomes

$$\begin{aligned} &\max_{\boldsymbol{y}_n\in\mathcal{Y}_n,n\in\mathcal{V}} \sum_{n\in\mathcal{V}} F_n(\boldsymbol{y}_n) \\ &\text{s.t.} \quad \sum_{n\in\mathcal{V}}\mathbf{g}_n(\boldsymbol{y}_n) \leq \mathbf{0}, \end{aligned} \tag{7}$$

where $F_n(\boldsymbol{y}_n)$ is the solution of the sub-problem in (6).

The sub-problem in (6) is a well-studied one which is typically solved with a master-slave dual method [6]. The dual problem for (6) can be written as

$$\min_{\mu_n\geq 0}\max_{\mathbf{x}_i\in\mathcal{X}_i,i\in\mathcal{S}_n} \sum_{i\in\mathcal{S}_n}\left(f_i(\mathbf{x}_i) - \boldsymbol{\mu}_n^\top\mathbf{h}_i(\mathbf{x}_i,\boldsymbol{y}_n)\right), \tag{8}$$

where the inner maximization can be solved in edge devices for fixed $\boldsymbol{\mu}_n$ since it requires only f_i and $\mathbf{h}_i$, and the outer minimization can be handled in server n via gradient descent after aggregation of the results passed by the edge devices, where this step does not require the knowledge of private f_i's. Specifically, the gradient descent iterations for the multiplier can be written as

$$\boldsymbol{\mu}_n(t+1) = \left[\boldsymbol{\mu}_n(t) + \gamma\sum_{i\in\mathcal{S}_n}\mathbf{h}_i(\mathbf{x}_i^*(\boldsymbol{\mu}_n(t)),\boldsymbol{y}_n)\right]^+, \tag{9}$$

where γ denotes the step-size and $[\cdot]^+$ denotes the projection onto nonnegative orthant. The new multiplier value is calculated at server node n and passed to the edge devices which are responsible for calculation of $\mathbf{x}_i^*(\boldsymbol{\mu}_n)$ for each $\boldsymbol{\mu}_n$ as

$$\mathbf{x}_i^*(\boldsymbol{\mu}_n) = \operatorname*{argmax}_{\mathbf{x}_i\in\mathcal{X}_i}\left\{f_i(\mathbf{x}_i) - \boldsymbol{\mu}_n^\top\mathbf{h}_i(\mathbf{x}_i,\boldsymbol{y}_n)\right\}. \tag{10}$$

In a nutshell, the dual algorithm works as follows. Each edge device updates their optimization variable according to (10), and pass the result to the server which aggregates these to update the multiplier via (9), where the result is passed back to the edge devices.[2] This message passing continues iteratively until a convergence criterion is satisfied. The typical analogy for this algorithm is with the law of supply and demand from economics, where the multiplier denotes the *price*. Edge devices calculate the optimal resource required for them with their private utilities given a price, whereas the price is updated in an aggregator based on the total demand (i.e., asked resources) and the supply.

After solving the sub-problems among servers and their corresponding edge devices, we can potentially apply the same algorithm to the network-wide problem (7), however, the aggregation entity does not exist in the architecture we consider in this paper. This is why we propose a method based on the consensus among server nodes.

The dual problem for (1) can be written as

$$\min_{\lambda}\min_{\mu}\max_{\boldsymbol{y}}\max_{\mathbf{x}} L(\boldsymbol{\theta}) = \min_{\lambda}\max_{\boldsymbol{y}}\min_{\mu}\max_{\mathbf{x}} L(\boldsymbol{\theta}),$$

where $\boldsymbol{\theta} = \{\mathbf{x},\boldsymbol{y},\boldsymbol{\lambda},\boldsymbol{\mu}\}$, and we exchange the order of the middle two iterations using minimax theorem[3] on the right hand side. Therefore, we obtain an optimization order from the local to the global variables. We write gradient steps for the $\boldsymbol{\lambda}$ and $\boldsymbol{y}$, where the sub-problem can be solved with (9) and (10).

Since the outer optimizations use the results of the inner ones, we highlight that multiple iterations of the inner

[2]If closed-form minimization is not possible in the edge devices, one can use primal-dual updates, in which optimal $\mathbf{x}_i$'s are also found via projected gradient descent iterations.

[3]We use compact convex sets as domains of the optimization variables.

optimization updates are needed before an iteration on the outer optimizations. Let us assume there are k updates of $\boldsymbol{\mu}_n$ before an update of $\boldsymbol{y}_n$ and $\boldsymbol{y}_n$ is updated k' times between the updates of $\boldsymbol{\lambda}$. We propose an algorithm which we call Decentralized Federated Network Utility Maximization (Decentralized Fed-NUM) in Algorithm 1.

Algorithm 1: Decentralized Fed-NUM

At edge device $i \in \mathcal{S}_n$: To compute $\mathbf{x}_i(t)$ each client for every $t = 1, 2, \ldots$, either solves

$$\mathbf{x}_i(t) = \underset{\mathbf{x}_i \in \mathcal{X}_i}{\operatorname{argmax}} \{f_i(\mathbf{x}_i) - \boldsymbol{\mu}_n(t)^\top \mathbf{h}_i\big(\mathbf{x}_i, \boldsymbol{y}_n\big(\lfloor t/k \rfloor\big)\big)\}$$

in closed-form or uses a projected gradient step:

$$\mathbf{x}_i(t) = \mathfrak{P}_{\mathcal{X}_i}\Big[\mathbf{x}_i(t-1) + \eta\Big(\nabla_{\mathbf{x}_i} f_i(\mathbf{x}_i(t-1)) - \boldsymbol{\mu}_n(t)^\top \nabla_{\mathbf{x}_i} \mathbf{h}_i\big(\mathbf{x}_i(t-1), \boldsymbol{y}_n\big(\lfloor t/k \rfloor\big)\big)\Big)\Big],$$

and sends $\mathbf{x}_i(t)$ to server n.

At server $n \in \mathcal{V}$: To update sub-problem dual variable $\boldsymbol{\mu}_n$ every time the edge devices send their solution:

$$\boldsymbol{\mu}_n(t+1) = \Big[\boldsymbol{\mu}_n(t) + \gamma \sum_{i \in \mathcal{S}_n} \mathbf{h}_i(\mathbf{x}_i(t), \boldsymbol{y}_n\big(\lfloor t/k \rfloor\big))\Big]^+.$$

To solve the network-wide problem, for every integer τ when $t = \tau k$:

$$\boldsymbol{y}_n(\tau+1) = \mathfrak{P}_{\mathcal{Y}_n}\Big[\boldsymbol{y}_n(\tau) - \beta\Big(\mu_n(\tau k) \nabla_{\boldsymbol{y}_n} \mathbf{h}_i(\mathbf{x}_i(\tau k), \boldsymbol{y}_n(\tau)) + \boldsymbol{\lambda}_n(\lfloor \tau/k' \rfloor) \nabla_{\boldsymbol{y}_n} \mathbf{g}_n(\boldsymbol{y}_n(\tau))\Big)\Big]$$

Let $\boldsymbol{\lambda}_n$ denote the local copy of server n of the multiplier $\boldsymbol{\lambda}$. For every integer j when $t = jkk'$:

$$\boldsymbol{\lambda}_n(j+1) = \Big[\sum_{n' \in \mathcal{N}_n} [\mathbf{W}]_{nn'} \boldsymbol{\lambda}_{n'}(j) + \alpha N \mathbf{g}_n(\boldsymbol{y}_n(jk'))\Big]^+$$

Here $\mathcal{N}_n$ is the neighborhood set of server n including itself, and $\mathbf{W}$ is the doubly-stochastic mixing matrix. $\alpha, \beta, \gamma, \eta > 0$ are step sizes. In addition, $\mathfrak{P}_{\mathcal{X}}$ denotes projection onto set $\mathcal{X}$, and $\nabla_{\mathbf{x}}$ denotes gradient on the $\mathbf{x}$ direction.

In the algorithm, we solve the sub-problem (6) iteratively, for both $\mathbf{x}_i$ and $\boldsymbol{\mu}_n$. Note that $\boldsymbol{y}_n\big(\lfloor t/k \rfloor\big)$ remains constant for k such iterations. For the network-wide problem, we use gradient updates where all the gradient values in this algorithm are found by taking gradients of the Lagrangian in (2) with respect to the corresponding variables. We then write gradient descent for minimization and ascent for maximization. The only non-traditional step taken here is for the updates of the multiplier $\boldsymbol{\lambda}$ which is the global variable in the system. This is because we do not have a central unit to gather all the information, perform the gradient update and broadcast it back. Instead, we have server nodes that are connected to each other over a network. If there was a central node, the update would be

$$\boldsymbol{\lambda}(j+1) = \Big[\boldsymbol{\lambda}(j) + \alpha \sum_{n \in \mathcal{V}} \mathbf{g}_n(\boldsymbol{y}_n(jk'))\Big]^+, \tag{11}$$

which requires all $\boldsymbol{y}_n$ values in an update. Instead, in the algorithm, we keep local copies of this dual variable in each node where we perform local iterations using local $\boldsymbol{y}_n$, and we use a consensus algorithm that provides inexact decentralized averaging. In the consensus step, each server n exchanges their local copies $\boldsymbol{\lambda}_n$ with the neighboring nodes $n' \in \mathcal{N}_n$. Assume a mixing matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ with $[\mathbf{W}]_{ij} = 0$ if $(i,j) \notin \mathcal{E}$ and $i \neq j$. Furthermore, $\mathbf{W} = \mathbf{W}^\top$, $\mathbf{W}\mathbf{1}_N = \mathbf{1}_N$ and $\rho(\mathbf{W} - \mathbf{1}_N \mathbf{1}_N^\top / N) < 1$, where $\rho(\cdot)$ is the spectral radius and $\mathbf{1}_N \in \mathbb{R}^N$ is the column vector of ones. Then, for this doubly-stochastic $\mathbf{W}$, we have $\lim_{\phi \to \infty} \mathbf{W}^\phi = \frac{1}{N}\mathbf{J}$, where $\mathbf{J} \in \mathbb{R}^{N \times N}$ with $[\mathbf{J}]_{ij} = 1$, $\forall i, j \in \{1, \ldots, N\}$. With this relationship, we can state that multiplication of a vector with the mixing matrix continuously (i.e., asymptotically) makes all the elements of the vector equal to the average of this vector. Therefore, we can argue that using the updates for the local copies $\boldsymbol{\lambda}_n$ with continuous mixing is equivalent to the use of the updates in (11). However, in order to gain time, we use only one mixing (exchange) round after each k' iterations of $\boldsymbol{y}_n$, i.e., we use one multiplication with mixing matrix, which makes the decentralized averaging inexact for the proposed algorithm. The choice of using only one mixing round lets other iterations use the updated value of $\boldsymbol{\lambda}_n$ earlier, however, on the other hand, creates sub-optimality due to the use of the inexact averaging.

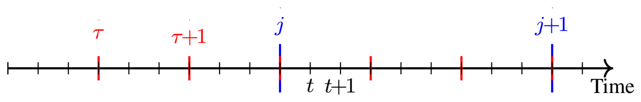

Fig. 3: Time-scale of updates for different iterations. The increasingly coarser time indices t, τ, and j index the updates of the local resource prices $\boldsymbol{\mu}$, the server resources $\boldsymbol{y}$, and the global price $\boldsymbol{\lambda}$, respectively.

In the proposed algorithm, we consider different time-scales for different iterations, where the local updates occur more frequently than the global ones. Fig. 3 shows the different time indices of the different iterations in an example where $k = k' = 3$. Executing the global iterations more sporadically gives chance to the local iterations to reach a point close to the optimum for fixed global variables, which reduces the gradient errors in iterations. This phenomenon is analyzed extensively with theoretical results in [18] for the hierarchical multi-layer NUM, where the effects of the choices of k and k' on the convergence are characterized.

The solution of the sub-problem uses a master-slave architecture, and therefore, it requires worker nodes to report their results to the master node, where the master node broadcasts updated dual variables. On the other hand, we use peer-to-peer exchanges in the solution of network-wide problem in order to provide a consensus for the global dual variable $\boldsymbol{\lambda}$. Fig. 4 illustrates the required message exchange on the architecture. The

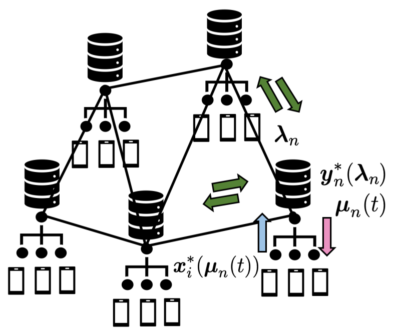
$\boldsymbol{\lambda}_n$
$\boldsymbol{y}_n^*(\boldsymbol{\lambda}_n)$
$\boldsymbol{\mu}_n(t)$
$\boldsymbol{x}_i^*(\boldsymbol{\mu}_n(t))$

Fig. 4: The sub-problem requires vertical exchanges whereas the network-wide problem requires exchanges among neighboring server nodes.

proposed algorithm combines the hierarchical NUM approach with a consensus algorithm for decentralization. We note that the choice of the consensus scheme can be replaced with many other methods with directed and undirected message exchanges.

To measure the benefits in terms of response time roughly, we compare a two-layer architecture where the end users interact directly with a central controller, with the proposed architecture where end users are equally split under the control of N servers (i.e., each server has the same number of users) connected through a graph. We assume there are two types of control channels: a high latency channel c, where all the communications occur in the two-layer architecture and where the communications among servers occur in the graph network, and a low latency channel c', where the intra-server communications occur among the edge devices and the corresponding servers. 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, υ for channel c and υ' 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 $\upsilon' < \upsilon$. Hence, in the two-layer systems, the latency for each update of the central controller is $\upsilon N|S|_n$. In the other system the total latency, instead, is $\Delta(\mathcal{G})\upsilon + kk'|\mathcal{S}_n|\upsilon'$, where the first term accounts for the total time it takes for servers to communicate, and the second term is due to the fact that there are kk' iterations at the bottom end-user layer between any two iterations on the top layer, each with average latency $|\mathcal{S}_n|\upsilon'$. Here, $\Delta(\mathcal{G})$ is the maximum degree of graph $\mathcal{G}$. Thus, we gain in latency per iteration if the ratio δ of these two numbers exceeds one, i.e.:

$$\delta \triangleq \frac{|\mathcal{S}_n|N}{\Delta(\mathcal{G}) + kk'|\mathcal{S}_n|\frac{\upsilon'}{\upsilon}} > 1. \tag{12}$$

IV. Numerical Performance Analysis

We consider a resource allocation problem with private utilities $f_i = w_i \log(1 + a_i x_i)$ of edge devices, where the weights w_i's are drawn from a uniform distribution between $[1, 2]$, and a_i's follow Rayleigh distribution with parameter 1. We assume

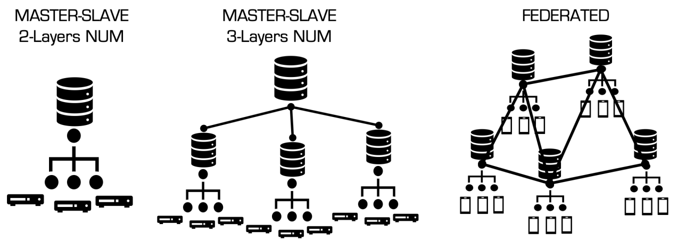

Fig. 5: Benchmarks for numerical evaluation

we have a total resource constraint as $\sum_{n\in\mathcal{V}}\sum_{i\in\mathcal{S}_n} x_i \le Z$, where Z denotes the total resource. This constraint can equivalently be rewritten as $\sum_{i\in\mathcal{S}_n} x_i \le y_n, \forall n \in \mathcal{V}$ (as local/server constraints in (1)) and $\sum_{n\in\mathcal{V}} y_n \le Z$ (as a global constraint in (1)) with the introduction of slack variables y_n's. Then, based on Alg. 1, we can write the update rules as

$$\begin{aligned}
x_i(t) &= \mathfrak{P}_{\mathcal{X}_i}[\dot{f}_i^{-1}(\mu_n(t))] \\
\mu_n(t+1) &= \left[\mu_n(t) + \gamma\Big(\sum_{i\in\mathcal{S}_n} x_i(t) - y_n\big(\lfloor t/k \rfloor\big)\Big)\right]^+ \\
y_n(\tau+1) &= \mathfrak{P}_{\mathcal{Y}_i}[y_n(\tau) + \beta(\mu_n(\tau k) - \lambda\big(\lfloor \tau/k' \rfloor\big))] \\
\lambda_n(j+1) &= \left[\sum_{n'\in\mathcal{N}_n}[\mathbf{W}]_{nn'}\lambda_{n'}(j) + \alpha(Ny_n(jk') - Z)\right]^+,
\end{aligned} \tag{13}$$

where $\dot{f}_i^{-1}$ is the inverse function of the derivative $\dot{f}_i$.

We set $|\mathcal{S}_n| = 20, \forall n \in \mathcal{V}$, $N = 10$ and $Z = 100$. The connection graph among servers is based on the Erdős-Rényi model, where the probability of being present is set 0.35 for all possible edges[4]. The weights of the doubly-stochastic mixing matrix are set via Metropolis weights:

$$[\mathbf{W}]_{nn'} = \begin{cases} 1/\max\{|\mathcal{N}_n|, |\mathcal{N}_{n'}|\} & \{n, n'\} \in \mathcal{E} \\ 1 - \sum_{n'\in\mathcal{N}_n\setminus\{n\}}[\mathbf{W}]_{nn'} & n = n' \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

These weights are symmetric and doubly-stochastic, i.e., the entries in the columns and rows add up to 1. Furthermore, they are easy to compute and suitable with the distributed implementation, where each node requires the size of the neighborhood set of its neighbors. This information can be obtained in two rounds of peer-to-peer communications: in the first round they can count their $|\mathcal{N}_n|$, and then share it with their neighbors in the second round. This simplicity is especially useful for time-varying graphs [19].

We compare three algorithms implemented in corresponding three architectures: 1) 2-Layer NUM, where all $|S|_n \times N = 200$ edge devices report to only one controller, without any computation in between. The classical dual algorithm is enough to solve the problem in this case with a single dual

[4]For simplicity, we consider each server has the same number of edge devices and we assign simple parameter values in general. However, more complex scenarios such as very unbalanced number of edge devices for servers require further attention in future studies.

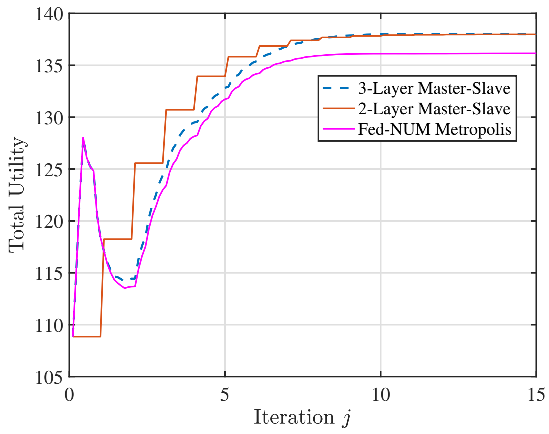

Fig. 6: Convergence speed in terms of dual variable iteration count

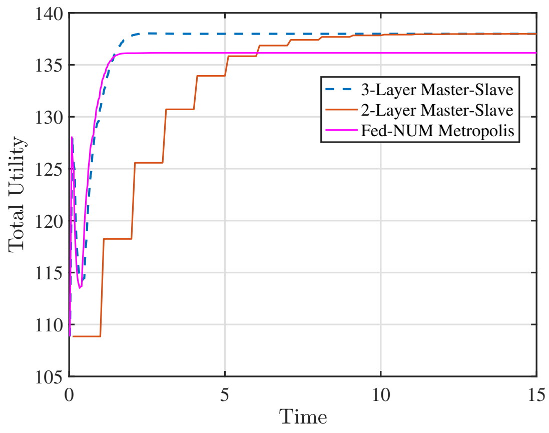

Fig. 7: Convergence speed including latency gains

variable that is updated in the controller. 2) 3-Layer NUM, where we have servers with their edge devices similar to the our consideration, however, there is also a central node managing the servers. This case uses (11) rather than message exchanges for consensus. 3) The proposed algorithm in Alg.1 with updates in (13). In Fig. 5, we illustrate the architectures that are considered for comparison.

Fig. 6 and Fig. 7 illustrate our simulation results for different architectures where $k = k' = 3, \upsilon/\upsilon' = 5$. In Fig. 6, we present the convergence behaviors when the dual variable $\boldsymbol{\lambda}$ iterations are aligned in the horizontal axis. As expected, 3-Layer Master-Slave and Fed-NUM iterations have similar trajectory due to the fact that they conduct similar lower-layer updates. However, there is a performance gap in terms of optimal utility that is caused by the inexact averaging of the Fed-NUM. In Fig. 7, the horizontal axis denotes time, and in this case, for Fed-NUM, we observe significant increase of speed compared to 2-Layer case, and a small increase compared to the 3-Layer case which requires a central controller. This speed can be increased further by decreasing $\Delta(\mathcal{G})$ as shown in (12), however, reducing connectivity results worse approximations for averaging in finite-time, and therefore, the optimality gap increases and there is a trade-off between speed vs. accuracy.

V. Conclusions

We have introduced a decentralized large-scale optimization algorithm for an architectural model defined as mesh network of star networks. The algorithm combines multi-layer NUM framework with a consensus step. We provide results stating a decentralized, private, fast implementation is possible with a small price from accuracy.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, 2016.

[2] 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.

[3] 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, 1998.

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

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

[6] 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, 2007.

[7] L. Ferrari, N. Karakoç, A. Scaglione, M. Reisslein, and A. Thyagaturu, “Layered cooperative resource sharing at a wireless SDN backhaul,” in *Proc. IEEE Int. Conf. on Communications Workshops (ICC Workshops)*, Kansas City, MO, May 2018, pp. 1–6.

[8] P. Shantharama, A. S. Thyagaturu, N. Karakoç, 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. 57 545–57 561, 2018.

[9] M. Wang, N. Karakoc, L. Ferrari, P. Shantharama, A. S. Thyagaturu, M. Reisslein, and A. Scaglione, “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, 2019.

[10] N. Karakoç, A. Scaglione, and A. Nedić, “Multi-layer decomposition of optimal resource sharing problems,” in *Proc. IEEE Conf. on Decision and Control (CDC)*, Miami Beach, FL, 2018, pp. 1–6.

[11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[12] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” *IEEE Trans. Autom. Control*, vol. 54, no. 1, p. 48, 2009.

[13] B. Johansson, T. Keviczky, M. Johansson, and K. H. Johansson, “Subgradient methods and consensus algorithms for solving convex optimization problems,” in *Proc. IEEE Conf. on Decision and Control (CDC)*, Cancún, Mexico, 2008, pp. 4185–4190.

[14] T.-H. Chang, A. Nedić, and A. Scaglione, “Distributed constrained optimization by consensus-based primal-dual perturbation method,” *IEEE Trans. Autom. Contr.*, vol. 59, no. 6, pp. 1524–1538, 2014.

[15] R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperation in networked multi-agent systems,” *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[16] L. Xiao and S. Boyd, “Fast linear iterations for distributed averaging,” *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[17] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, “Stochastic gradient push for distributed deep learning,” *arXiv preprint arXiv:1811.10792*, 2018.

[18] N. Karakoc, A. Scaglione, A. Nedic, and M. Reisslein, “Multi-layer decomposition of network utility maximization problems,” *IEEE/ACM Transactions on Networking, submitted.*

[19] L. Xiao, S. Boyd, and S. Lall, “Distributed average consensus with time-varying metropolis weights,” *Automatica*, 2006.