2018 IEEE Conference on Decision and Control (CDC)
Miami Beach, FL, USA, Dec. 17-19, 2018

Multi-layer Decomposition of Optimal Resource Sharing Problems

Nurullah Karakoç, Anna Scaglione and Angelia Nedić
School of Electrical, Computer and Energy Engineering, Arizona State University
nkarakoc, anna.scaglione, angelia.nedich@asu.edu

Abstract—**We describe a distributed framework for resource sharing problems that we face in communications, microeconomics and various networking applications. In particular, we consider a hierarchical multi-layer decomposition for network utility maximization (NUM), where functionalities are assigned to different layers. The proposed methodology creates solutions having central management and distributed computations. The technique aims to respond to the dynamics of the network by decreasing the communication cost, while shifting more computational load to the edges of the network. The main contribution of this work is the provision of a detailed analysis under the assumption that the network changes are in the same time-scale with the convergence time of the algorithms used for local computations. For this scenario, assuming strong concavity and smoothness of the users' objective functions, we present convergence rates for each layer.**

I. Introduction

In this paper, we revisit the classic problem of allocating resources in a network with many users by decomposing it into interacting layers, with a central mechanism that distributes resources among these different users based on time-varying demand profiles and channel conditions. The main example inspiring this work is the study we initiated in [1] of a possible distributed implementation of a Software Defined Network (SDN) controller placed at the Internet backhaul, serving different wireless access networks (such as eNB, Wi-Fi AP) and also serving different service providers. Our decomposition, however, finds other applications in, for instance, the optimal distribution of a product among warehouses for several retailers [2], or the optimal caching of content on Internet servers [3].

In 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 creates a costly communication bottleneck; yet, there are potentially great benefits from harmonizing the way all of these different operators and end-users are served, rather than allocating resources statically.

Resource allocation problems are well-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 [4], [5]. Within the NUM literature, a variety of decentralized solutions (surveyed in [6]) are presented with different assumptions and setups. These solutions commonly use the method of parameter exchange between edges and the central control to build a decentralized implementation as a result of mathematical decomposition of the problem (see [7], [8] for various decomposition techniques). An important issue in these solutions is the time scales of the computations and the demand changes and the control signalling overhead when the network has a large footprint. The common assumption is that the convergence time of the proposed methods is much shorter than rate at which the demand, or the channel conditions, change; in other words the parameters of the problem are assumed to be static during the execution of the methods. This assumption, referred to as *time-scale separation*, separates the optimization procedure from the dynamics of the underlying system and works well in environments that are relatively stable or that allow for fast computations and control channels messages. In [9], 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 arrival and service rates of a Markovian queueing model.

To overcome the communication bottleneck, and consider also the dynamics of the system parameters, in this paper we propose a hierarchical decomposition approach to the resource allocation problem clustering nodes at each layer of the hierarchy and introducing slack variables to represent the resource managed by nodes at a certain intermediate layer. The idea behind the approach is very simple. With the slack variables, we create multiple layers between the end-users and the central controller. At the lowest layers are the users which are clustered under higher layers nodes that act as distributors for the lower layer elements in their cluster.

Our technical contributions are as follows: We present a distributed algorithm based on projected gradient iterations for multi-layer decomposition of resource allocation problems, and we show a linear convergence rate for constant step size under some concavity and smoothness assumptions on the utility functions. In addition, we evaluate the performance of the proposed methods with numerical examples.

II. Problem formulation

Consider a network of N users, each with objective to maximize his/her own (scalar) utility function. However, the users decisions are coupled through a common resource,

The work has been partially supported by the NSF grant CCF-1717391, the ONR grant no. N00014-12-1-0998 and the NSF grant NeTS: 1716121.

where the amount of total resource is limited by R_{tot}. We can formulate this resource allocation problem as follows:

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

where s denotes 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 transpose operation. There can be additional constraints depending on the application such as the link capacity constraints in communication networks, which can be easily included in our framework. Replacing x_s by y_s^1, we define the L-layer decomposition problem as follows:

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

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 a node n in the layer $\ell+1$ above layer ℓ. Naturally, $N_1 = N$ and $N_L = 1$. We assume 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 3-layer architecture.

The nodes in higher layers act as distributors of resources to the nodes in their 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 reflects naturally modules and boundaries that are present in a communication architecture. Hence, the computational resources that can tackle the intermediate problems are already in place, they simply currently not co-optimize their share of resources dynamically. Also, geographical clustering is natural in network services as well, and in this case the clusters represent main control points in each region.

The main benefit of creating sub-optimization procedures is that the resulting decentralized algorithm reduces communication costs, and it can react faster to the changes in the demand profile, by reducing the delay in the parameter exchanges[1]. Each layer acts as a central decision maker of the resource allocation for the one lower layer element without waiting for the full convergence of an optimizing method employed in the system. In order to do that, different layers use different time scales in their computations. Lower layers have faster time scales than the higher layers in a solution procedure and the decision making procedure is shifted to the edges for faster response times. Our goal in the next section is to analyze the convergence properties of such an algorithm.

[1]We note that delay refers to the first reaction time of the network here, not the total delay between top and bottom layers.

III. Distributed Algorithm and its Convergence

We make the following assumptions:

a.1) Utility functions U_s are increasing, strictly concave and twice 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 $x_s \in \mathcal{I}_s$.

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

Before discussing our multi-layer decomposition algorithm in Section III-B, we first review the conventional two layer decomposition method (see e.g. [5]) 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.:

$$\begin{aligned} L(\mathbf{x}, \lambda) &= \sum_{s=1}^{N} U_s(x_s) - \lambda\left(\sum_{s=1}^{N} x_s - R_{tot}\right) \\ &= \sum_{s=1}^{N} \left(U_s(x_s) - \lambda x_s\right) + \lambda R_{tot}, \end{aligned} \tag{3}$$

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

$$g(\lambda) = \max_{x_s \in \mathcal{I}_s, 1 \leq s \leq N} L(\mathbf{x}, \lambda) = \sum_{s=1}^{N} f_s(\lambda) + \lambda R_{tot}, \tag{4}$$

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

$$f_s(\lambda) = \max_{x_s \in \mathcal{I}_s} \{U_s(x_s) - \lambda x_s\} \tag{5}$$

and the dual problem is

$$\min_{\lambda \geq 0} g(\lambda). \tag{6}$$

The decentralized implementation finds the optimal x_s of the subproblems in (5) for a given λ, while the optimum value of λ is found via a gradient projection algorithm. Specifically, the iterations for λ are:

$$\begin{aligned} \lambda(t+1) &= \left(\lambda(t) - \gamma \frac{\partial g(\lambda(t))}{\partial \lambda}\right)^+ \\ &= \left(\lambda(t) + \gamma\left(\sum_{s=1}^{N} x_s^*(\lambda(t)) - R_{tot}\right)\right)^+, \end{aligned} \tag{7}$$

where $\gamma > 0$ denotes the step size, $(\cdot)^+$ represents the projection onto the nonnegative orthant, while $x_s^*(\lambda)$ denotes the solution to subproblem (5) for a given λ, i.e.,

$$x_s^*(\lambda) = [\dot{U}_s^{-1}(\lambda)]_{m_s}^{M_s}, \tag{8}$$

where $\dot{U}_s^{-1}$ is the inverse function of $\dot{U}_s$ which denotes derivative of 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 (8) and passes this value to the central controller. Then, the controller updates the price according to (7) and passes this value to the users. This process continues iteratively until some convergence criterion is satisfied. The typical analogy is with 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/hers 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 reported in [5] with a slightly different problem setup including link constraints. Note that, by using (8), we can find second derivative:

$$\frac{\partial^2 g(\lambda)}{\partial \lambda^2} = -\sum_{s=1}^{N} \frac{\partial x_s^*(\lambda)}{\partial \lambda}, \tag{9}$$

where

$$\frac{\partial x_s^*(\lambda)}{\partial \lambda} = \begin{cases} \frac{1}{\ddot{U}_s(x_s^*(\lambda))}, & \text{for } \dot{U}_s(m_s) \geq \lambda \geq \dot{U}_s(M_s), \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

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

$$0 < -1/\ddot{U}_s(x_s^*) \leq \alpha_s < \infty \qquad \text{for all } s. \tag{11}$$

Next, we define $\overline{\alpha} = \max_s(\alpha_s)$. With minor changes in the proof in [5], under assumptions **a.1** and **a.2**, it can be shown that $g(\lambda)$ has Lipschitz gradients with a constant $N\overline{\alpha}$ [5]. Under assumptions **a.1–a.2**, the dual problem (6) has a nonempty solution set Λ^*, and we have:

Theorem 1: The method (7) 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

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

where g^* denotes the dual optimal value and $\lambda(0) \geq 0$ is the initial iterate value[2].

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

$$g(\lambda(t)) - g^* \leq \frac{N\overline{\alpha}}{2t}\left\|\lambda(0) - \lambda^*\right\|^2. \tag{12}$$

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

B. A 3-Layer Distributed Algorithm and Convergence Rates

Assume we have the 3-layer decomposition structure for a resource sharing problem as in Fig. 1. In the architecture, we have one controller on the top layer which has fixed total resource $R_{tot} = Z$ 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 is denoted by y_o. In the bottom layer, we have end-users connected to operators where $\mathcal{S}^o$ represents the set of end-users connected to operator o. To highlight the difference among the intermediate layer and the top and bottom layers, we will refer to $y_1^L = R_{tot} = Z$ and use directly x_s rather than y_s^1, while for the 2nd layer we will omit the layer superscript $\ell = 2$ and simply refer to the variables as y_o, while we will name μ_o the dual variables of the constraints between second and first layer and name λ the dual variable between layer 2 and 3. More specifically, the optimization problem becomes:

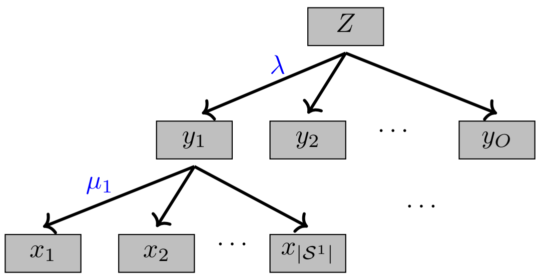

Fig. 1. 3-Layer Decomposition Structure

$$\max_{\mathbf{x}\in\mathcal{I},\mathbf{y}\geq\mathbf{0}} \sum_{o=1}^{O}\sum_{s\in\mathcal{S}^o} U_s(x_s) \tag{13}$$
$$\text{s.t.} \sum_{o=1}^{O} y_o \leq Z, \qquad \sum_{s\in\mathcal{S}^o} x_s \leq y_o, \quad o = 1,\ldots,O,$$

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:

$$L(\mathbf{x},\mathbf{y},\boldsymbol{\mu},\lambda) = \sum_{o=1}^{O}\sum_{s\in\mathcal{S}^o} U_s(x_s) - \sum_{o=1}^{O}\mu_o\Bigg(\sum_{s\in\mathcal{S}^o} x_s - y_o\Bigg)$$
$$-\lambda\Bigg(\sum_{o=1}^{O} y_o - Z\Bigg), \tag{14}$$

where $\boldsymbol{\mu}$ is the column vector with entries μ_o. Moreover, μ_o, as mentioned before, is the Lagrange multiplier associated with the feasibility constraint $\sum_{s\in\mathcal{S}^o} x_s \leq y_o$ for all $o = 1,\ldots,O$ which couple layers 1 and 2, while 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:

$$\begin{aligned} g(\lambda,\boldsymbol{\mu}) &= \max_{\mathbf{x}\in I,\mathbf{y}\geq 0} L(\mathbf{x},\mathbf{y},\boldsymbol{\mu},\lambda) \\ &= \sum_{o=1}^{O}\sum_{s\in\mathcal{S}^o} f_s(\mu_o) + \sum_{o=1}^{O} h_o(\mu_o,\lambda) + \lambda Z, \end{aligned} \tag{15}$$

where

$$f_s(\mu_o) = \max_{x_s\in\mathcal{I}_s}\{U_s(x_s) - \mu_o x_s\}, \tag{16}$$

$$h_o(\mu_o,\lambda) = \max_{y_o\geq 0} y_o(\mu_o - \lambda), \tag{17}$$

[2]Here, we slightly modify the convergence theorem in [5] to get a converge rate by using [10, Thm.3.1].

and the dual problem is

$$\min_{\lambda\geq 0,\boldsymbol{\mu}\geq 0} g(\lambda,\boldsymbol{\mu}). \tag{18}$$

Let $\boldsymbol{\theta} = (\mathbf{x}^\top, \mathbf{y}^\top, \boldsymbol{\mu}^\top, \lambda)^\top$. For the dual problem, due to the minimax properties:

$$\min_{\lambda\geq 0}\min_{\boldsymbol{\mu}\geq 0}\max_{\mathbf{y}\geq 0}\max_{\mathbf{x}\in\mathcal{I}} L(\boldsymbol{\theta}) = \min_{\lambda\geq 0}\max_{\mathbf{y}\geq 0}\min_{\boldsymbol{\mu}\geq 0}\max_{\mathbf{x}\in\mathcal{I}} L(\boldsymbol{\theta}), \tag{19}$$

where we use separability over variables of the both primal and dual problems in the first step and minimax theorem to substitute 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 which leads to local resource allocation subproblems for the operators that can be also solved distributedly.

Unlike the two layer case, in the case of three or more layers to establish our results regarding the convergence of the algorithm we need to focus on the cases where the optimal allocations are interior points of the feasible region, i.e., we assume that there exist $m_s < x_s^*(\mu_o, y_o, \lambda) < M_s, \forall s, o$, which by (8) imply that $\dot{U}_s(m_s) > \mu_o > \dot{U}_s(M_s), \forall s, o$. For this assumption to be true, the following should hold:

$$\sum_{o=1}^{O}\sum_{s\in\mathcal{S}^o} m_s < Z < \sum_{o=1}^{O}\sum_{s\in\mathcal{S}^o} M_s, \tag{20a}$$

$$\underline{y}_o = \sum_{s\in\mathcal{S}^o} m_s < y_o^* < \sum_{s\in\mathcal{S}^o} M_s = \overline{y}_o, \tag{20b}$$

$$\underline{\mu}_o = \max_{s\in\mathcal{S}^o} \dot{U}_s(M_s) < \mu_o^* < \min_{s\in\mathcal{S}^o} \dot{U}_s(m_s) = \overline{\mu}_o, \tag{20c}$$

$$\underline{\lambda} = \max_{s} \dot{U}_s(M_s) < \lambda^* < \min_{s} \dot{U}_s(m_s) = \overline{\lambda}, \tag{20d}$$

where the bounds (20a), (20b) are necessary for feasibility, and the dual variable bounds (20c), (20d) should hold as a result of (8) for the interior point solutions x_s^*. The benefit of having this assumption is basically escaping from having zero terms in (10). This assumption is relaxed in the follow-up work [11].

In order to describe the algorithm updates, it is important to notice that multiple iterations are necessary in general to solve for the optimum resource allocation for any given value of the dual variables that is associated with the constraint. 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 the latter is updated k' times before the global price λ is updated. We can define the state of the optimization $\boldsymbol{\theta}(t)$ as follows:

$$\boldsymbol{\theta}(t) = \left(\mathbf{x}^\top(t), \mathbf{y}^\top(\lfloor t/k\rfloor), \boldsymbol{\mu}^\top(t), \lambda(\lfloor t/kk'\rfloor)\right)^\top. \tag{21}$$

With this in mind, we can write:

$$x_s(t) = \dot{U}_s^{-1}(\mu_o(t)), \tag{22}$$

$$\begin{aligned}\mu_o(t+1) &= \left[\mu_o(t) - \gamma_o''\frac{\partial L(\boldsymbol{\theta}(t))}{\partial \mu_o}\right]_{\underline{\mu}_o}^{\overline{\mu}_o} \qquad (23)\\ &= \left[\mu_o(t) + \gamma_o''\left(\sum_{s\in\mathcal{S}_o} x_s(t) - y_o(\lfloor t/k\rfloor)\right)\right]_{\underline{\mu}_o}^{\overline{\mu}_o},\end{aligned}$$

and for the intermediate layer we can write:

$$\begin{aligned}y_o(i+1) &= \left[y_o(i) + \gamma_o'\frac{\partial L(\boldsymbol{\theta}(ik))}{\partial \mu_o}\right]_{\underline{y}_o}^{\overline{y}_o} \qquad (24)\\ &= \left[y_o(i) + \gamma_o'\big(\mu_o(ik) - \lambda(\lfloor i/k'\rfloor)\big)\right]_{\underline{y}_o}^{\overline{y}_o},\end{aligned}$$

$$\begin{aligned}\lambda(j+1) &= \left[\lambda(j) - \gamma\frac{\partial L(\boldsymbol{\theta}(jkk'))}{\partial \lambda}\right]_{\underline{\lambda}}^{\overline{\lambda}} \qquad (25)\\ &= \left[\lambda(j) - \gamma\big(y_o(jk') - Z\big)\right]_{\underline{\lambda}}^{\overline{\lambda}},\end{aligned}$$

where $\gamma, \gamma_o', \gamma_o''$ represent different constant step sizes.

Therefore, the algorithm works as follows. In the bottom layer, the end-users find the optimal amount of resources for a fixed local price μ_o, a fixed operator resource supply y_o and a fixed global price λ, by using (22). The value calculated is then passed to the operators they connect to. Then, the operator o sets a new local prices based on its total supply y_o and the optimal resource requested by the nodes in the lower layer, according to (23). 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 supply amount y_o according to (24) and uses the iteration outcome to calculate a new optimum local price. This continues until convergence to a value close to the optimum y_o^* for the given λ. Then, similarly, by using these y_o^* values and the total resource 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 parameters 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, and these operators that 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.

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

a.4) $\quad -\ddot{U}_s(x_s) \leq 1/\beta_s < \infty \quad$ for all $\quad x_s \in I_s$. (26)

Let:

$$\alpha_o^{(1)} = \max_{s\in\mathcal{S}^o}(\alpha_s), \qquad \beta_o^{(1)} = \min_{s\in\mathcal{S}^o}(\beta_s), \tag{27}$$

$$\alpha^{(2)} = \max_{o}(|\mathcal{S}^o|\alpha_o^{(1)}), \quad \beta^{(2)} = \min_{o}(|\mathcal{S}^o|\beta_o^{(1)}). \tag{28}$$

Assuming the problem has a nonempty solution set under conditions (20), we obtain:

Theorem 2: Let **a.1–a.4** hold. For fixed $\lambda(j)$ and $y_o(i)$, the method (23) with constant step size $\gamma_o'' = \frac{2}{|\mathcal{S}^o|(\alpha_o^{(1)}+\beta_o^{(1)})}$ produces iterates $\mu_o(t)$ that converge to the optimum solution $\mu_o^* := \mu_o^*(\lambda, y_o)$ with a linear rate, i.e., for each $o = 1, \ldots, O$:

$$\|\mu_o(t) - \mu_o^*\| \leq \left(\frac{\alpha_o^{(1)} - \beta_o^{(1)}}{\alpha_o^{(1)} + \beta_o^{(1)}}\right)^t \|\mu_o(0) - \mu_o^*\|. \tag{29}$$

Assuming that $\mu_o(t)$ is converging to μ_o^* in bottom layer iterations, for a fixed $\lambda(j)$, the method (24) with constant step size $\gamma_o' = \frac{2|\mathcal{S}^o|\alpha_o^{(1)}\beta_o^{(1)}}{\alpha_o^{(1)}+\beta_o^{(1)}}$ produces iterates $y_o(i)$ converging to the optimum solution $y_o^* := y_o^*(\lambda)$ with a linear rate, i.e., for each $o = 1, \ldots, O$:

$$\|y_o(i) - y_o^*\| \leq \left(\frac{\alpha_o^{(1)} - \beta_o^{(1)}}{\alpha_o^{(1)} + \beta_o^{(1)}}\right)^i \|y_o(0) - y_o^*\| . \qquad (30)$$

Assuming that $y_o(i)$ is converging to y_o^* in middle layer iterations, the method (25) with constant step size $\gamma = \frac{2}{O(\alpha^{(2)}+\beta^{(2)})}$ produces iterates $\lambda(j)$ that converge to the optimum solution λ^* with a linear rate, i.e.,

$$\|\lambda(j) - \lambda^*\| \leq \left(\frac{\alpha^{(2)} - \beta^{(2)}}{\alpha^{(2)} + \beta^{(2)}}\right)^j \|\lambda(0) - \lambda^*\| . \qquad (31)$$

Proof Omitted for brevity[3]. ■

It can be seen that 3-layer algorithm convergence rate is the same as that of the 2-layer algorithm convergence rate when lower layer delays are negligible compared to the delays of the communication links with the main controller, (i.e., when middle layer is close to the end-users).

C. Extensions to L-layer Decomposition

Let $\boldsymbol{y}_\ell = (y_1^\ell, \ldots, y_{N_\ell}^\ell)^\top$, $\ell = 1, \ldots, L$, $U(\boldsymbol{y}_1) = \sum_{s=1}^N U_s(y_s^1)$ 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 $\boldsymbol{\lambda}_\ell = (\lambda_1^\ell, \ldots, \lambda_{N_{\ell+1}}^\ell)^\top$, and:

$$\boldsymbol{\theta} = \begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{\lambda} \end{pmatrix} \qquad (32)$$

the Lagrangian of the problem (2) can be written as:

$$\begin{aligned} L(\boldsymbol{\theta}) &= U(\boldsymbol{y}_1) - \sum_{\ell=1}^{L-1} (\mathbf{A}_\ell \boldsymbol{y}_\ell - \boldsymbol{y}_{\ell+1})^\top \boldsymbol{\lambda}_\ell \\ &= U(\boldsymbol{y}_1) - \boldsymbol{y}_1^\top \mathbf{A}_1^\top \boldsymbol{\lambda}_1 \qquad (33) \\ &\quad + \sum_{\ell=2}^{L-1} \boldsymbol{y}_\ell^\top (\boldsymbol{\lambda}_{\ell-1} - \mathbf{A}_\ell^\top \boldsymbol{\lambda}_\ell) + \boldsymbol{\lambda}_{L-1} \boldsymbol{y}_1^L, \end{aligned}$$

where $\boldsymbol{\lambda}_{L-1}$ and $\boldsymbol{y}_1^L$ are scalars since there is only one element in the top layer, and the objective of the dual problem can be written as:

$$g(\boldsymbol{\lambda}) = f(\boldsymbol{\lambda}_1) + \sum_{\ell=2}^{L-1} h_\ell(\boldsymbol{\lambda}_{\ell-1}, \boldsymbol{\lambda}_\ell) + \lambda_{L-1} y_1^L, \qquad (34)$$

where:

$$f(\boldsymbol{\lambda}_1) = \max_{\boldsymbol{y}_1 \in \mathcal{I}} \big(U(\boldsymbol{y}_1) - \boldsymbol{y}_1^\top \mathbf{A}_1^\top \boldsymbol{\lambda}_1\big) \qquad (35)$$

$$h_\ell(\boldsymbol{\lambda}_{\ell-1}, \boldsymbol{\lambda}_\ell) = \max_{\boldsymbol{y}_\ell \geq \mathbf{0}} \big(\boldsymbol{y}_\ell^\top (\boldsymbol{\lambda}_{\ell-1} - \mathbf{A}_\ell^\top \boldsymbol{\lambda}_\ell)\big). \qquad (36)$$

[3]It will be presented in [11] along with extension to the L-layer case.

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 variables $\boldsymbol{\lambda}_\ell$, and there are also k_ℓ updates of the dual variables $\boldsymbol{\lambda}_\ell$ for every $\boldsymbol{y}_{\ell+1}$. Therefore, with respect to the index of the fastest updates t, at each 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 follows:

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

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 index i as follows:

$$\begin{aligned} \boldsymbol{\lambda}_\ell(i+1) &= \left[\boldsymbol{\lambda}_\ell(i) - \mathbf{G}_\ell \frac{\partial L(\boldsymbol{\theta}(i p_\ell k_\ell'))}{\partial \boldsymbol{\lambda}_\ell}\right]_{\underline{\boldsymbol{\lambda}}_\ell}^{\overline{\boldsymbol{\lambda}}_\ell} \qquad (37) \\ &= \big[\boldsymbol{\lambda}_\ell(i) + \mathbf{G}_\ell\big(\mathbf{A}_\ell \boldsymbol{y}_\ell(i k_\ell') - \boldsymbol{y}_{\ell+1}(\lfloor i/k_\ell \rfloor)\big)\big]_{\underline{\boldsymbol{\lambda}}_\ell}^{\overline{\boldsymbol{\lambda}}_\ell}, \end{aligned}$$

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

$$\begin{aligned} \boldsymbol{y}_\ell(j+1) &= \left[\boldsymbol{y}_\ell(j) + \mathbf{G}_\ell' \frac{\partial L(\boldsymbol{\theta}(j p_\ell))}{\partial \boldsymbol{y}_\ell}\right]_{\underline{\boldsymbol{y}}_\ell}^{\overline{\boldsymbol{y}}_\ell} \qquad (38) \\ &= \big[\boldsymbol{y}_\ell(j) + \mathbf{G}_\ell'\big(\boldsymbol{\lambda}_{\ell-1}(j k_{\ell-1}) - \mathbf{A}_\ell^\top \boldsymbol{\lambda}_\ell(\lfloor j/k_\ell' \rfloor)\big)\big]_{\underline{\boldsymbol{y}}_\ell}^{\overline{\boldsymbol{y}}_\ell}, \end{aligned}$$

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

$$y_s^1(t) = [\dot{U}_s^{-1}(\lambda_n^1(t))]_{m_s}^{M_s}, \quad s \in \mathcal{S}_1^n. \qquad (39)$$

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 $\overline{\lambda}_n^1 = \min_{s \in \mathcal{S}_1^n} \dot{U}_s(m_s)$ and $\underline{\lambda}_n^1 = \max_{s \in \mathcal{S}_1^n} \dot{U}_s(M_s)$. 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 $\underline{y}_n^2 = \sum_{s \in \mathcal{S}_1^n} m_s$.

The result of Theorem 2 can be extended to the L-layer decomposition for problem (2) in a straightforward manner. As shown, there is a pattern in the convergence rates and one can easily write convergence rates of all layers similarly for both dual variable updates and allocated resource updates.

We can introduce a recursive definition of:

$$\alpha_n^{(\ell)} = \max_{s \in \mathcal{S}_\ell^n}(|\mathcal{S}_{\ell-1}^s|\alpha_s^{(\ell-1)}), \ \beta_n^{(\ell)} = \min_{s \in \mathcal{S}_\ell^n}(|\mathcal{S}_{\ell-1}^s|\beta_s^{(\ell-1)}), \ (40)$$

for $\ell = 2, .., L-1$ where $\alpha_n^{(1)}$ and $\beta_n^{(1)}$ are defined in (27). With respect to the dual variables λ_n^ℓ associated to the constraints of the ℓth layers' resources $\sum_{s \in \mathcal{S}_\ell^n} y_s^\ell \leq y_n^{\ell+1}$ for $n = 1, \ldots, N_{\ell+1}$, when the upper layer parameters are fixed and assuming that $y_s^{\ell*}(\lambda_n^\ell)$ 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 $n = 1, \ldots, N_{\ell+1}$:

$$\left\|\lambda_n^\ell(i) - \lambda_n^{\ell *}\right\| \leq \left(\frac{\alpha_n^{(\ell)} - \beta_n^{(\ell)}}{\alpha_n^{(\ell)} + \beta_n^{(\ell)}}\right)^i \left\|\lambda_n^\ell(0) - \lambda_n^{\ell *}\right\|. \quad (41)$$

For the resources of 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 $s = 1, \ldots, N_\ell$:

$$\left\|y_s^\ell(j) - y_s^{\ell *}\right\| \leq \left(\frac{\alpha_s^{(\ell-1)} - \beta_s^{(\ell-1)}}{\alpha_s^{(\ell-1)} + \beta_s^{(\ell-1)}}\right)^j \left\|y_s^\ell(0) - y_s^{\ell *}\right\|. \quad (42)$$

IV. Numerical Examples

In our numerical test, we have $N = 80$ end-users and their utility functions are of the form $U_s(x) = w_s \log(1+x)$, with weights w_s drawn from a uniform distribution on $[1, 2]$. We have a dynamic system such that w_s's are randomly drawn every 4 seconds. We run two different implementations: (1) the 2-Layer algorithm where the sources are directly communicating with the central distributor, and (2) the 3-Layer algorithm where we have $O = 4$ operators each connected to 20 sources, i.e., $|\mathcal{S}^o| = 20, o = 1, 2, 3, 4$. We assume that main delay cause is the communication link with the central distributor also in the 3-Layer algorithm, i.e., the operators are located close to the edge nodes. The termination threshold is selected as 10^{-5} where we allow a maximum of 100 iterations for lower layers, (i.e., $k' = k = 100$). We also set $R_{tot} = 800$, and $I_s = [0, 800]$ for all s as total resource and feasibility boundaries, respectively. The step sizes are $\gamma = 2.5 \cdot 10^{-5}$, $\gamma' = 400$ and $\gamma'' = 10^{-4}$.

In the 2-Layer algorithm, the end-users start using the optimum x_s after the convergence criterion is met, since the current iterates can be infeasible. Before convergence, they use previous optimal rates that are produced for the previous realization of w_s's. Similarly, for the middle layer resources (y_o's) in the 3-Layer algorithm, one should wait for global convergence to use them. However, the feasible y_o values of the previous layer is used at the beginning of each time slot (each 4s), and the end-users allocations and local prices converge in this local market. Therefore, these end-users allocations can be used without waiting for full convergence. With the multi-layer algorithm, the system reacts much faster to the underlying changes in the network. As shown in Fig. 2, 3-Layer algorithm reacts immediately whereas in 2-Layer algorithm first reaction time is basically the full convergence time. Before full convergence of the system, even though 3-Layer algorithm does not reach optimum, it provides significant increase in the total utility. We observe the same convergence rate for both algorithms (as discussed at the end of Sec. III-B).

V. Conclusions

In order to decrease message passing costs and reaction time of distributed price-based sharing algorithms, multi-layer decomposition structure looks promising since it has the same global convergence rate and error bounds characteristics as the conventional methods. The multi-layer decomposition reacts significantly faster to the demand change and shifts the computation to the edges. Due to the space constraints, optimality bounds with the projected erroneous gradient methods in dynamic resource allocation where the optimal result we seek changes with time will be discussed in [11].

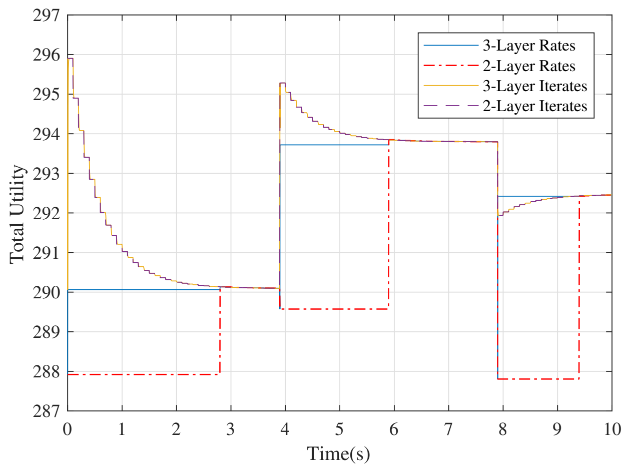

Fig. 2. Comparison of algorithms in a dynamic system.

References

[1] 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. on Communications Workshops (ICC Workshops)*, May 2018, pp. 1–6.

[2] J.-B. Sheu, "A novel dynamic resource allocation model for demand-responsive city logistics distribution operations," *Transportation Research Part E: Logistics and Transportation Review*, vol. 42, no. 6, pp. 445–472, Nov. 2006.

[3] 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.

[4] 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.

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

[6] 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.

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

[8] 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.

[9] 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.

[10] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[11] N. Karakoc, A. Scaglione, A. Nedic, and M. Reisslein, "Multi-layer decomposition of network utility maximization problems," *in preparation*.