Distributed Cross-Layer Optimization in Wireless Networks: A Second-Order Approach

Jia Liu* Cathy H. Xia† Ness B. Shroff* Hanif D. Sherali‡
* Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210
† Department of Integrated Systems Engineering, The Ohio State University, Columbus, OH 43210
‡ Grado Department of Industrial Systems Engineering, Virginia Tech, Blacksburg, VA 24061

***Abstract*—Due to the rapidly growing scale and heterogeneity of wireless networks, the design of distributed cross-layer optimization algorithms has received significant interest from the networking research community. So far, the standard distributed cross-layer approach in the literature is based on the first-order Lagrangian dual decomposition and the subgradient method, which suffers from a slow convergence rate. In this paper, we make the first known attempt to develop a distributed Newton's method, which is *second-order* and enjoys a *quadratic* convergence rate. However, due to the inherent interference in wireless networks, the Hessian matrix of the cross-layer problem has a *non-separable* structure. As a result, developing a distributed second-order algorithm is far more difficult than its counterpart for wireline networks. Our main contributions in this paper are two-fold: i) For a special network setting where all links mutually interfere, we derive closed-form expressions for the Hessian inverse, which further yield a distributed Newton's method; ii) For general wireless networks where the interference relationships are arbitrary, we propose a double matrix-splitting scheme, which also leads to a distributed Newton's method. Collectively, these results create a new theoretical framework for distributed cross-layer optimization in wireless networks. More importantly, our work contributes to a potential second-order paradigm shift in wireless networks optimization theory.**

I. Introduction

The proliferation of mobile communication devices (e.g., smartphones, tablets, etc.) has been accompanied by a rapid growth in scale and heterogeneity of wireless networks. As a result, distributed cross-layer algorithms have received significant interest from the wireless networking research community in recent years. In the literature, the standard approach for distributed optimization in wireless networks is based on the Lagrangian dual decomposition framework and the subgradient method (LD-SG), which is primarily due to its elegant cross-layer implementations (see, e.g., [1] and references therein). The LD-SG framework is also intimately linked to the celebrated throughput-optimal "back-pressure" algorithm [2], which has led to a large number of routing and scheduling schemes for wireless networks (see, e.g., [3]–[6]). However, despite its theoretical and engineering appeals, the performance of LD-SG is not satisfactory in practice. Being a first-order approach in nature (search directions are based on the first-order supports of the dual function), the subgradient method suffers from a slow convergence rate and is sensitive to step-size choices [7]. Due to these limitations, in this paper, we consider designing a distributed Newton's method for cross-layer optimization in wireless networks. The fundamental philosophy of this approach is that, being a *second-order* method, a distributed Newton's algorithm exploits both the gradient and Hessian information in determining search directions. Hence, an appropriately designed distributed Newton's method would also enjoy the powerful *quadratic rate of convergence* as in classical Newton type methods [7], [8].

However, developing second-order distributed algorithms for wireless networks is highly challenging and, to our knowledge, results in this area remain elusive. Due to a very different problem structure in wireless networks, techniques used for developing distributed second-order algorithms in wireline networks [9]–[11] cannot be directly applied (see Section II for more detailed discussions). Generally speaking, in a distributed second-order algorithm, computing the primal and dual search directions typically requires decomposing the inverses of the Hessian matrix and a weighted Laplacian matrix (weighted by the Hessian inverse), and then distributing each piece to each network entity (i.e., a node or a link). Unfortunately, unlike wireline networks for which the Hessian is (block) diagonal (see [11]), the Hessian's structure is *non-separable* due to the inherent interference in wireless networks. What is worse is that, not only are both the Hessian and weighted Laplacian inversions cumbersome in large-scale wireless networks, the obtained inverses also have no sparsity structure in general. Hence, distributed computations of the Hessian and weighted Laplacian inversion problems in wireless networks are far more difficult than their counterparts in wireline networks.

The key contribution of this paper is that we successfully develop a series of new second-order techniques to overcome all of the above difficulties in wireless networks. Hence, our work can be viewed as the first building block towards the development of an analytical foundation for cross-layer design that provides second-order convergence speed. The main technical contributions of this paper are as follows:

- We first consider a special network setting where every two links mutually interfere and cannot transmit simultaneously (e.g., CSMA networks, cellular uplink/downlink, etc.). In this case, by exploiting the special "arrow-head" sparsity structure in the Hessian (cf. Eq. (18)), we prove that the

This work has been supported in part by the Army Research Office MURI Award W911NF-08-1-0238 and NSF grants CNS-1065136, CNS-1012700, IIS-0916440, ECCS-1232118, and CMMI-0969169.

inverse of the Hessian matrix is also an "arrow-head" matrix and can be computed in *closed-form*, thus significantly reducing the computational complexity. More importantly, the derived closed-form expression of each entry in the Hessian inverse naturally leads to a distributed implementation.

- We next consider general wireless networks where the interference relationships are arbitrary. In the general case, since the Hessian inverse is non-sparse and deriving its closed-form expressions is intractable, we propose to iteratively compute the Hessian inverse and the weighted Laplacian inverse by a new *double matrix-splitting* technique. This double matrix-splitting scheme can be parameterized for convergence speed tuning and, more importantly, implemented in a distributed fashion.
- We offer interesting insights and networking interpretations for our proposed distributed algorithms, as well as the connections with and differences from first-order approaches. This further advances our understanding of second-order approaches in wireless network optimization theory.

To the best of our knowledge, this paper is the first work that develops a distributed second-order method for cross-layer optimization in wireless networks. Collectively, our results serve as an important first step in providing a cross-layer solution for wireless networks using second-order techniques. The remainder of this paper is organized as follows. In Section II, we review related work in the literature, putting our work in a comparative perspective. Section III introduces the network model and problem formulation. Section IV develops the principal components of our distributed Newton's method. Section V presents some relevant numerical results, and Section VI concludes this paper.

II. Related Work

Since distributed second-order methods for wireless networks have not been investigated in the literature, the works being surveyed herein are for wireline networks only. Historically, second-order methods for network optimization (both centralized and distributed) date back to the 1980s, including, e.g., a centralized projected Newton's method for multi-commodity flow problems [12] and a distributed conjugate gradient direction method for solving pure minimum cost flow routing problems [13]. These early attempts all employed gradient projections to identify feasible search directions. In contrast, most of the recent works in this area [9]–[11], [14]–[16] are based on the interior-point approach [17] due to its superior efficiency in both theory and practice. One of the first applications of an interior-point based second-order method was developed for the pure flow control problem (with fixed routing) [14], where Zymnis *et al.* proposed a centralized truncated-Newton's primal-dual algorithm. Bickson *et al.* [15], [16] also studied the same problem and designed a distributed algorithm based on the Gaussian belief propagation technique to avoid direct Hessian inversion [18]. Alternatively, Wei *et al.* [10] approached the same distributed flow control problem and computed the Hessian inverse based on an iterative matrix-splitting scheme. A distributed Newton's method was also developed for the pure minimum cost routing problem (with fixed source rates) by Jadbabaie *et al.* in [9], where they proposed a consensus-based local averaging scheme to iteratively compute the Hessian inverse and established its convergence using spectral graph theory [19]. Finally, in our previous work [11], we showed that, through a suitable reformulation that exposes a block diagonal structure in the Hessian matrix, a distributed Newton's method can be developed for the more complex joint multi-path routing and flow control problem by generalizing the matrix-splitting idea in [11]. However, we point out that none of the aforementioned techniques can be directly applied to wireless networks due to a completely different Hessian matrix structure (cf. Eq. (18)), and our development of the distributed Newton's method for wireless networks cross-layer optimization is completely new.

III. Network Model and Problem Formulation

We first introduce our notation used in this paper. We use boldface to denote matrices and vectors. For a matrix $\mathbf{A}$, $\mathbf{A}^T$ denotes the transpose of $\mathbf{A}$. $\mathrm{Diag}\{\mathbf{A}_1,\ldots,\mathbf{A}_N\}$ represents the block diagonal matrix with matrices $\mathbf{A}_1,\ldots,\mathbf{A}_N$ on its main diagonal. Also, $\mathrm{diag}\{\mathbf{A}\}$ represents the vector containing the main diagonal entries of $\mathbf{A}$. We let $(\mathbf{A})_{ij}$ represent the entry in the i-th row and j-th column of matrix $\mathbf{A}$ and let $(\mathbf{v})_m$ represent the m-th entry of vector $\mathbf{v}$. We let $\mathbf{I}_K$ denote the K-dimensional identity matrix, and let $\mathbf{1}_K$ and $\mathbf{0}_K$ denote the K-dimensional vectors whose elements are all ones and zeros. We let $\mathbf{e}_K^{(k)}$ denote the k-th vector in the natural basis of $\mathbb{R}^K$ (i.e., the k-th entry is "1" and other entries are "0").

Network layer model. In this paper, a wireless network is represented by a directed graph, denoted by $\mathcal{G}=\{\mathcal{N},\mathcal{L}\}$, where $\mathcal{N}$ and $\mathcal{L}$ are the sets of nodes and links, respectively. We assume that $\mathcal{G}$ is connected. The cardinalities of the sets $\mathcal{N}$ and $\mathcal{L}$ are $|\mathcal{N}|=N$ and $|\mathcal{L}|=L$, respectively. We use the so-called *node-arc incidence matrix* (NAIM) [20] $\mathbf{A}\in\mathbb{R}^{N\times L}$ to represent the network topology of $\mathcal{G}$. Let $\mathrm{Tx}(l)$ and $\mathrm{Rx}(l)$ denote the transmitting and receiving nodes of link l, respectively. The entries of $\mathbf{A}$ are thus defined as follows:

$$(\mathbf{A})_{nl}=\begin{cases}1, & \text{if } n=\mathrm{Tx}(l),\\ -1, & \text{if } n=\mathrm{Rx}(l),\\ 0, & \text{otherwise.}\end{cases} \tag{1}$$

In the network, different source nodes send independent data to their intended destination nodes, potentially through multi-path and multi-hop routing. Suppose that there are F sessions in the network, representing F different commodities. We denote the source and destination nodes of session f, $1\le f\le F$, as $\mathrm{Src}(f)$ and $\mathrm{Dst}(f)$, respectively. The source flow rate of session f is denoted by a scalar $s_f\in\mathbb{R}_+$. For session f, we use a *source–destination vector* vector $\mathbf{b}_f\in\mathbb{R}^N$ to represent the supply–demand relationship of session f. More specifically, the entries in $\mathbf{b}_f$ are defined as follows:

$$(\mathbf{b}_f)_n=\begin{cases}1, & \text{if } n=\mathrm{Src}(f),\\ -1, & \text{if } n=\mathrm{Dst}(f),\\ 0, & \text{otherwise.}\end{cases} \tag{2}$$

For every link l, we let $x_l^{(f)} \geq 0$ represent the flow amount of session f on link l. We assume that the network is a flow-balanced system, i.e., the following flow-balanced constraints hold at each node:

$$\begin{aligned}
\sum_{l\in\mathcal{O}(n)} x_l^{(f)} - \sum_{l\in\mathcal{I}(n)} x_l^{(f)} &= s_f, \qquad \text{if } n = \mathrm{Src}(f),\\
\sum_{l\in\mathcal{O}(n)} x_l^{(f)} - \sum_{l\in\mathcal{I}(n)} x_l^{(f)} &= 0, \qquad \text{if } n \neq \mathrm{Src}(f), \mathrm{Dst}(f),\\
\sum_{l\in\mathcal{O}(n)} x_l^{(f)} - \sum_{l\in\mathcal{I}(n)} x_l^{(f)} &= -s_f, \quad \text{if } n = \mathrm{Dst}(f),
\end{aligned}$$

where $\mathcal{O}(n)$ and $\mathcal{I}(n)$ represent the sets of outgoing and incoming links at node n, respectively. We let $\mathbf{x}^{(f)} \triangleq [x_1^{(f)}, \ldots, x_L^{(f)}]^T \in \mathbb{R}_+^L$ denote the *routing vector* for session f across all links. Using the notation $\mathbf{A}$, $\mathbf{b}_f$, and $\mathbf{x}^{(f)}$, the above flow-balanced constraints can be compactly written as:

$$\mathbf{A}\mathbf{x}^{(f)} - s_f\mathbf{b}_f = \mathbf{0}, \quad \forall f = 1, 2, \ldots, F. \tag{3}$$

Note that in (3), $\mathbf{A}$ is not of full row rank (because all columns sum up to zero). To eliminate the redundant rows in $\mathbf{A}$, we let $\mathbf{A}^{(f)} \in \mathbb{R}^{(N-1)\times L}$ be obtained by deleting from $\mathbf{A}$ the row corresponding to the node $\mathrm{Dst}(f)$. It is easy to verify that $\mathbf{A}^{(f)}$ is of full row rank [20]. Also, we let $\widetilde{\mathbf{b}}^{(f)} \in \mathbb{R}^{N-1}$ be obtained by deleting from $\mathbf{b}_f$ the entry corresponding to the node $\mathrm{Dst}(f)$. Accordingly, we rewrite (3) as:

$$\mathbf{A}^{(f)}\mathbf{x}^{(f)} - s_f\widetilde{\mathbf{b}}^{(f)} = \mathbf{0}, \quad \forall f = 1, 2, \ldots, F. \tag{4}$$

Link layer model. In this paper, we adopt the following collision-based interference model at the link layer: In a given time instant, due to the shared nature of wireless media, only a subset of links can be activated simultaneously without interfering with each other. To model this, we let $\mathcal{C} \triangleq \{\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(I)}\}$ denote the set of all possible interference-free link rate vectors, where $\mathbf{c}^{(i)} \triangleq [C_1^{(i)}, \ldots, C_L^{(i)}]^T \in \mathbb{R}_+^L$. In $\mathbf{c}^{(i)}$, if $C_{l_1}^{(i)} > 0$ and $C_{l_2}^{(i)} > 0$ for some $l_1, l_2 \in \mathcal{L}$, then it implies that links l_1 and l_2 do not interfere with each other and are both activated. Under this model, selecting a link rate vector from $\mathcal{C}$ in each time instant is equivalent to activating a subset of interference-free links. For simplicity, in this paper, we do not consider channel variations (i.e., $C_l^{(i)}$ is time-varying due to fading and/or mobility of the nodes) when selecting the subset of interference-free links. Such "opportunism" of exploiting channel state information (CSI) will be left for our future study.

Now, we let $\Lambda \triangleq \mathrm{Co}(\mathcal{C}) \subset \mathbb{R}_+^L$ denote the link capacity region under the interference-free restriction, where $\mathrm{Co}(\cdot)$ represents the convex-hull operation. Then, a necessary condition for the network to be stable is that the flow routing vectors satisfy $\sum_{f=1}^F \mathbf{x}^{(f)} \in \Lambda$. Further, it is well-known that the convex-hull operation in Λ can be achieved through a standard time-sharing argument [1], [2]. Thus, we let t_i represent the fraction of time during which link rate vector $\mathbf{c}^{(i)}$ is selected, where t_i satisfies $0 \leq t_i \leq 1$ and $\sum_{i=1}^I t_i = 1$. With time-sharing, the capacity of link l under the interference-free restriction can be computed as $C_l = \sum_{i=1}^I t_i C_l^{(i)}$. Then, the aforementioned stability condition can be written explicitly as:

$$\sum_{f=1}^F x_l^{(f)} \leq \sum_{i=1}^I t_i C_l^{(i)}, \quad \forall l = 1, \ldots, L, \tag{5}$$

where t_i, $\forall i$, are decision variables. We remark that (5) is a necessary condition on the feasibility of average flow rates, and can be viewed as a relaxation of the instantaneous link capacity constraint where time-sharing is not allowed (i.e., replacing t_i by $\varphi_i^\tau \in \{0, 1\}$ in each time instant τ, where $\varphi_i^\tau = 1$ if $\mathbf{c}^{(i)}$ is selected in τ or 0 otherwise). Hence, the solutions obtained via (5) may be infeasible under the instantaneous link capacity constraint. But to enable the development of second-order methods while drawing useful insights for future scheduling schemes design, we choose to work with the constraints in (5) in this paper. We note, however, that the solution obtained via (5) is indeed achievable under time-sharing.

Problem formulation. We associate a utility function $U_f(s_f) : \mathbb{R}_+ \to \mathbb{R}$ with each session f. The overall network utility is given by $\sum_{f=1}^F U_f(s_f)$. We also assume that the utility functions U_f are strictly concave, monotonically increasing, twice continuously differentiable, and reversely self-concordant (see [8] for the definition of self-concordance). Our objective is to maximize the overall network utility. Putting together the models described earlier, we can formulate the cross-layer optimization (CLO) problem as follows:

CLO:

$$\begin{aligned}
\text{Maximize} \quad & \sum_{f=1}^F U_f(s_f)\\
\text{subject to} \quad & \mathbf{A}^{(f)}\mathbf{x}^{(f)} - s_f\widetilde{\mathbf{b}}^{(f)} = \mathbf{0}, && \forall f = 1, \ldots, F,\\
& \sum_{f=1}^F x_l^{(f)} \leq \sum_{i=1}^I t_i C_l^{(i)}, && \forall l = 1, \ldots, L,\\
& \sum_{i=1}^I t_i = 1,\\
& x_l^{(f)} \geq 0, \ \forall f, l; \quad s_f \geq 0, \ \forall f; \quad t_i \geq 0, \ \forall i.
\end{aligned}$$

Note that Problem CLO is a convex program and can be solved in the Lagrangian dual domain with a zero duality gap [7], [8]. Moreover, due to the separable structure of the dual function, Problem CLO can be solved distributedly by the dual decomposition and subgradient optimization (LD-SG) framework (see [1] or [21, Appendix A] for an overview). However, as mentioned earlier, the convergence performance of LD-SG is unsatisfactory. In what follows, we will investigate a new distributed second-order method to solve Problem CLO.

IV. A Distributed Newton's Method

In this section, we first reformulate Problem CLO to facilitate the second-order design of our distributed Newton's method in Section IV-A. Then, we investigate its Hessian matrix structure in Section IV-B. The distributed computations

of the primal Newton directions and the dual variables are presented in Sections IV-C and IV-D, respectively.

A. Problem Reformulation and Interior-Point Based Distributed Newton's Method

We start by reformulating Problem CLO using the interior-point framework. Following the standard interior-point approach [17], we apply a logarithmic barrier function to all inequality constraints and then accommodate them in the objective function. As a result, the augmented objective function (to be minimized) can be written as follows:

$$f_\mu(\mathbf{y}) = -\mu \sum_{f=1}^{F} U_f(s_f) - \sum_{l=1}^{L} \log\left(\sum_{i=1}^{I} t_i C_l^{(i)} - \sum_{f=1}^{F} x_l^{(f)}\right) - \sum_{f=1}^{F} \log(s_f) - \sum_{l=1}^{L}\sum_{f=1}^{F} \log(x_l^{(f)}) - \sum_{i=1}^{I} \log(t_i), \quad (6)$$

where $\mathbf{y} \triangleq \left[s_1 \cdots s_F \middle| x_1^{(1)} \cdots x_1^{(F)} \middle| \cdots \middle| x_L^{(1)} \cdots x_L^{(F)} \middle| t_1 \cdots t_I\right]^T$ groups all variables. In (6), $\mu > 0$ is a parameter that is used to track the central path in the interior-point method as $\mu \to \infty$ [8]. Moreover, we let

$$\mathbf{M} \triangleq \left[\begin{array}{c:ccc:c} & \mathbf{A}_1 & \cdots & \mathbf{A}_L & \\ \mathbf{B} & & & & \mathbf{1}_I^T \end{array}\right] \in \mathbb{R}^{[(N-1)F+1]\times[(L+1)F+I]},$$

where $\mathbf{B}$ and $\mathbf{A}_l$ are defined as $\mathbf{B} \triangleq \text{Diag}\{\widetilde{\mathbf{b}}^{(1)}, \ldots, \widetilde{\mathbf{b}}^{(F)}\}$, and $\mathbf{A}_l \triangleq \text{Diag}\{-\mathbf{a}_l^{(1)}, \ldots, -\mathbf{a}_l^{(F)}\}$, and where in the definition of $\mathbf{A}_l$, the vector $\mathbf{a}_l^{(f)}$ is the l-th column in the matrix $\mathbf{A}^{(f)}$ in Problem CLO (i.e., $\mathbf{A}^{(f)} = [\mathbf{a}_1^{(f)}, \mathbf{a}_2^{(f)}, \ldots, \mathbf{a}_L^{(f)}]$). Then, we can reformulate Problem CLO as follows:

$$\textbf{R-CLO:} \quad \begin{array}{ll} \text{Minimize} & f_\mu(\mathbf{y}) \\ \text{subject to} & \mathbf{M}\mathbf{y} = \mathbf{e}_{\bar{n}}^{(\bar{n})}, \end{array} \quad (7)$$

where $\bar{n} = (N-1)F + 1$. In $f_\mu(\mathbf{y})$, note that as $\mu \to \infty$, the original objective function of Problem CLO dominates the barrier functions, and hence the solution of Problem R-CLO approaches that of Problem CLO asymptotically. Further, since μ can be increased exponentially (e.g., letting $\mu_k = 2^k$), it suffices to focus on a second-order solution to the $f_\mu(\mathbf{y})$ problem in order to achieve a second-order convergence speed.

Now, we solve $f_\mu(\mathbf{y})$ by applying the (centralized) Newton's method, which is a second-order algorithm. Starting from an initial feasible solution $\mathbf{y}^0$, the centralized Newton's method iteratively searches for an optimal solution as follows:

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \pi^k \Delta\mathbf{y}^k, \quad (8)$$

where $\pi^k > 0$ is a positive step-size. In (8), $\Delta\mathbf{y}^k$ denotes the primal Newton direction, which is the solution to the following linear equation system obtained by deriving the Karush-Kuhn-Tucker (KKT) system of the second-order approximation of $f_\mu(\mathbf{y})$ [7], [8]:

$$\begin{bmatrix} \mathbf{H}_k & \mathbf{M}^T \\ \mathbf{M} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{y}^k \\ \mathbf{w}^k \end{bmatrix} = -\begin{bmatrix} \nabla f_\mu(\mathbf{y}^k) \\ \mathbf{0} \end{bmatrix}, \quad (9)$$

where $\mathbf{H}_k \triangleq \nabla^2 f_\mu(\mathbf{y}^k) \in \mathbb{R}^{(L+1)F\times(L+1)F}$ is the Hessian matrix of $f_\mu(\mathbf{y})$ at $\mathbf{y}^k$, and the vector $\mathbf{w}^k \in \mathbb{R}^{(N-1)F+1}$ contains the dual variables for the constraint $\mathbf{M}\mathbf{y} = \mathbf{e}_{\bar{n}}^{(\bar{n})}$ at the k-th iteration. Here, the entries in $\mathbf{w}^k$ are arranged as $[(\mathbf{w}_k^{(1)})^T, \ldots, (\mathbf{w}_k^{(F)})^T, w_k]^T$, where w_k is the dual variable associated with the time-sharing constraint $\sum_{i=1}^{I} t_i = 1$, and $\mathbf{w}_k^{(f)}$ is in the form of

$$\mathbf{w}_k^{(f)} \triangleq \left[w_1^{(f)}, \ldots, w_{\text{Dst}(f)-1}^{(f)}, w_{\text{Dst}(f)+1}^{(f)}, \ldots, w_N^{(f)}\right]^T. \quad (10)$$

Note that in (10), we have dropped the iteration index k within $[\cdot]$ for notational simplicity. For the same reason, in the rest of the paper, the iteration index k will be dropped whenever such an omission does not cause confusion. Also, we let $w_{\text{Dst}(f)}^{(f)} \equiv 0$, for all f. It can be readily verified that the coefficient matrix of the linear equation in (9) is nonsingular. Therefore, the primal Newton direction $\Delta\mathbf{y}^k$ and the dual variables $\mathbf{w}^k$ can be uniquely determined by solving (9). However, solving for $\Delta\mathbf{y}^k$ and $\mathbf{w}^k$ simultaneously via (9) requires global information and is difficult to be decentralized.

The first key step towards designing a distributed Newton's method is to solve (9) in an *alternative* fashion as follows:

$$\Delta\mathbf{y}^k = -\mathbf{H}_k^{-1}(\nabla f_\mu(\mathbf{y}^k) + \mathbf{M}^T\mathbf{w}^k), \quad (11)$$

$$\mathbf{w}^k = (\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T)^{-1}(-\mathbf{M}\mathbf{H}_k^{-1}\nabla f_\mu(\mathbf{y}^k)). \quad (12)$$

Hence, given $\mathbf{y}^k$, we can first compute $\mathbf{w}^k$ from (12). With $\mathbf{w}^k$, we can solve for $\Delta\mathbf{y}^k$ from (11). Then, $\Delta\mathbf{y}^k$ can be used in (8) (along with an appropriate step-size π^k) to determine the next primal feasible solution $\mathbf{y}^{k+1}$. However, as we shall see later, computing $\mathbf{H}_k^{-1}$ and $(\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T)^{-1}$ (which is the Laplacian matrix [19] weighted by $\mathbf{H}_k^{-1}$) remains difficult due to the *non-separable* structure of $\mathbf{H}_k$ and requires global information. This is in stark contrast to those optimization problems for wireline networks [9]–[11], where the Hessian matrices are (block) diagonal and their distributed inversion computations are much easier.

B. The Structure of the Hessian Matrix

To see the coupled and non-separable structure of $\mathbf{H}_k$, we evaluate the first and second partial derivatives of $f_\mu(\mathbf{y})$, for which the non-zero ones are:

$$\frac{\partial f_\mu}{\partial s_f} = -\mu U_f'(s_f) - \frac{1}{s_f}, \qquad \frac{\partial^2 f_\mu}{\partial s_f^2} = -\mu U_f''(s_f) + \frac{1}{s_f^2},$$

$$\frac{\partial f_\mu}{\partial x_l^{(f)}} = \frac{1}{\delta_l} - \frac{1}{x_l^{(f)}}, \qquad \frac{\partial^2 f_\mu}{\partial (x_l^{(f)})^2} = \frac{1}{\delta_l^2} + \frac{1}{(x_l^{(f)})^2},$$

$$\frac{\partial^2 f_\mu}{\partial x_l^{(f_1)} \partial x_l^{(f_2)}} = \frac{1}{\delta_l^2} \qquad \frac{\partial f_\mu}{\partial t_i} = -\sum_{l=1}^{L}\left(\frac{C_l^{(i)}}{\delta_l}\right) - \frac{1}{t_i},$$

$$\frac{\partial^2 f_\mu}{\partial t_i^2} = \sum_{l=1}^{L} \frac{(C_l^{(i)})^2}{\delta_l^2} + \frac{1}{t_i^2}, \quad \frac{\partial^2 f_\mu}{\partial t_{i_1} \partial t_{i_2}} = \sum_{l=1}^{L} \frac{C_l^{(i_1)} C_l^{(i_2)}}{\delta_l^2},$$

$$\frac{\partial^2 f_\mu}{\partial x_l^{(f)} \partial t_i} = -\frac{C_l^{(i)}}{\delta_l^2},$$

where $\delta_l \triangleq \sum_{i=1}^{I} t_i C_l^{(i)} - \sum_{f=1}^{F} x_l^{(f)}$ represents the *unused link capacity* of link l. For convenience, we use a vector

$$\mathbf{c}_l \triangleq [C_l^{(1)}, \ldots, C_l^{(I)}]^T \in \mathbb{R}^I \quad (13)$$

to group the capacity values of the l-th link in each of the I link rate vectors. We further define the following matrices:

$$\mathbf{S} \triangleq \mathrm{Diag}\left\{-\mu U_1''(s_1) + \frac{1}{s_1^2}, \ldots, -\mu U_F''(s_F) + \frac{1}{s_F^2}\right\}, \quad (14)$$

$$\mathbf{X}_l \triangleq \mathrm{Diag}\left\{\frac{1}{(x_l^{(1)})^2}, \ldots, \frac{1}{(x_l^{(F)})^2}\right\} + \frac{1}{\delta_l^2}\mathbf{1}_F\mathbf{1}_F^T, \quad \forall l, \quad (15)$$

$$\mathbf{C}_l \triangleq -\frac{1}{\delta_l^2}\mathbf{1}_F\mathbf{c}_l^T, \quad \forall l, \quad (16)$$

$$\mathbf{T} \triangleq \mathrm{Diag}\left\{\frac{1}{t_1^2}, \ldots, \frac{1}{t_I^2}\right\} + \sum_{l=1}^{L}\frac{1}{\delta_l^2}\mathbf{c}_l\mathbf{c}_l^T. \quad (17)$$

Then, it can be verified that the Hessian matrix $\mathbf{H}_k$ has the following "arrow-head" structure:

$$\mathbf{H}_k = \left[\begin{array}{ccccc} \mathbf{S} & & & & \\ & \mathbf{X}_1 & & & \mathbf{C}_1 \\ & & \ddots & & \vdots \\ & & & \mathbf{X}_L & \mathbf{C}_L \\ & \mathbf{C}_1^T & \cdots & \mathbf{C}_L^T & \mathbf{T} \end{array}\right]. \quad (18)$$

Remark 1. *It is insightful to compare the structure of $\mathbf{H}_k$ with that in [11]. Due to the absence of the time-sharing component, the Hessian in [11, Section V-C] is block diagonal and exactly the same as the principal submatrix $\mathrm{Diag}\{\mathbf{S}, \mathbf{X}_1, \ldots, \mathbf{X}_L\}$ in (18). In this paper, however, the coupling between the routing x-variables and the time-sharing t-variables yields two non-zero "bands" consisting of the $\mathbf{C}_l$-matrices. Thus, the block diagonal structure is destroyed in (18) after incorporating the time-sharing component. Hence, $\mathbf{H}_k^{-1}$ needs not be sparse even though $\mathbf{H}_k$ itself has certain sparsity structure. Also, due to the "arrow-head" structure in (18), finding closed-form expressions for $\mathbf{H}_k^{-1}$ is intractable in general. Fortunately, as will be shown in later sections, the arrow-head structure still provides some unique features that can be exploited to iteratively and distributedly compute $\mathbf{H}_k^{-1}$. Furthermore, under an interesting special case, this arrow-head structure leads to closed-form expressions for $\mathbf{H}_k^{-1}$ and reveals many interesting networking insights.*

C. Distributed Computation of Primal Newton Directions

In this subsection, we first consider a special collision-based network setting in which every two links in the network mutually interfere. In this special network setting, we derive closed-form expressions for $\mathbf{H}_k^{-1}$, which further lead to a fully distributed computational scheme for primal Newton directions. Next, we propose a matrix-splitting based technique for computing the primal Newton directions for general wireless network settings.

1) Primal Newton Directions: A Special Network Setting. We now consider a special network setting where every two links in the network mutually interfere with each other. This special setting is interesting in that it is a relevant model for many collision-based network architectures within one common interference domain (e.g., CSMA networks, cellular uplinks/downlinks, or dense ad hoc networks, etc.). In this network setting, due to the pairwise interference relationship, it is apparent that there can only be L link rate vectors (i.e., $I = L$), each of which has only one active link. In this case, without loss of generality, we can redefine $\mathbf{c}_l$ in (13) as follows:

$$\mathbf{c}_l = [0 \cdots 1 \cdots 0]^T \in \mathbb{R}^L \equiv \mathbf{e}_L^{(l)}, \quad (19)$$

i.e., the only non-zero entry "1" appears at the l-th position. For simplicity, we assume that each active link has a unit capacity. The extension to cases with arbitrary positive capacity values is straightforward, but at the expense of more complex notation. With this new notion of $\mathbf{c}_l$ and (17), we have

$$\mathbf{T} = \mathrm{Diag}\left\{\left(\frac{1}{t_1^2} + \frac{1}{\delta_1^2}\right), \ldots, \left(\frac{1}{t_L^2} + \frac{1}{\delta_L^2}\right)\right\}. \quad (20)$$

We point out that the diagonal structure of $\mathbf{T}$ in (20) will play a critical role in deriving the closed-form expressions for $\mathbf{H}_k^{-1}$ and the primal Newton directions.

Our key idea to compute $\mathbf{H}_k^{-1}$ in the special setting is to rewrite $\mathbf{H}_k$ in a *decomposition structure* to enable the application of the Sherman–Morrison–Woodbury (SMW) matrix inversion lemma [7]. In what follows, we outline the key steps. First, we define two new vectors as follows:

$$\widetilde{\mathbf{e}}_l \triangleq [\mathbf{0}_F^T, \ldots, \mathbf{1}_F^T, \ldots, \mathbf{0}_F^T, \mathbf{0}_L^T]^T \in \mathbb{R}^{(L+1)F+L},$$
$$\widetilde{\mathbf{c}}_l \triangleq [\mathbf{0}_F^T, \ldots, \mathbf{0}_F^T, \mathbf{c}_l^T]^T \in \mathbb{R}^{(L+1)F+L}.$$

In the definition of $\widetilde{\mathbf{e}}_l$, the vector $\mathbf{1}_F$ appears at the $(l+1)$-st block. Then, it can be readily verified that the "arrow-head" structure of $\mathbf{H}_k$ in (18) can be decomposed into a block diagonal matrix coupled with a rank-$2L$ update as follows:

$$\mathbf{H}_k = \mathrm{Diag}\{\mathbf{S}, \mathbf{X}_1, \ldots, \mathbf{X}_L, \mathbf{T}\} + \sum_{l=1}^{L}\left(-\frac{1}{\delta_l^2}\right)\widetilde{\mathbf{e}}_l\widetilde{\mathbf{c}}_l^T + \sum_{l=1}^{L}\left(-\frac{1}{\delta_l^2}\right)\widetilde{\mathbf{c}}_l\widetilde{\mathbf{e}}_l^T, \quad (21)$$

where rank-L updates $\sum_{l=1}^{L}(-\frac{1}{\delta_l^2})\widetilde{\mathbf{e}}_l\widetilde{\mathbf{c}}_l^T$ and $\sum_{l=1}^{L}(-\frac{1}{\delta_l^2})\widetilde{\mathbf{c}}_l\widetilde{\mathbf{e}}_l^T$ yield the vertical and horizontal bands in (18), respectively. For convenience, we define the following matrices:

$$\mathbf{D} \triangleq \mathrm{Diag}\{\mathbf{S}, \mathbf{X}_1, \ldots, \mathbf{X}_L, \mathbf{T}\},$$
$$\mathbf{U} \triangleq \left[\tfrac{\widetilde{\mathbf{e}}_1}{-\delta_1}, \cdots, \tfrac{\widetilde{\mathbf{e}}_L}{-\delta_L}, \tfrac{\widetilde{\mathbf{c}}_1}{-\delta_1}, \cdots, \tfrac{\widetilde{\mathbf{c}}_L}{-\delta_L}\right],$$
$$\mathbf{V} \triangleq \left[\tfrac{\widetilde{\mathbf{c}}_1}{\delta_1}, \cdots, \tfrac{\widetilde{\mathbf{c}}_L}{\delta_L}, \tfrac{\widetilde{\mathbf{e}}_1}{\delta_1}, \cdots, \tfrac{\widetilde{\mathbf{e}}_L}{\delta_L},\right]^T.$$

Then, Eq. (21) can be compactly written as $\mathbf{H}_k = \mathbf{D} + \mathbf{U}\mathbf{V}$. It then follows from the SMW matrix inversion lemma that

$$\mathbf{H}_k^{-1} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{U}(\mathbf{I} + \mathbf{V}\mathbf{D}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{D}^{-1}. \quad (22)$$

Now, we consider the computation of each term in the right-hand-side (RHS) of (22). First, thanks to the *block diagonal structure* of $\mathbf{D}$, we have that $\mathbf{D}^{-1}$ is simply $\mathrm{Diag}\left\{\mathbf{S}^{-1}, \mathbf{X}_1^{-1}, \ldots, \mathbf{X}_L^{-1}, \mathbf{T}^{-1}\right\}$. Further, due to the same structures as their counterparts in [11], $\mathbf{S}^{-1}$ and $\mathbf{X}_l^{-1}$ can be computed in closed-form by using Lemma 4 and Theorem 5 in [11]. Also, since $\mathbf{T}$ is diagonal, we immediately have that $\mathbf{T}^{-1}$ can be computed distributedly (link-wise) in closed-form as: $\mathbf{T}^{-1} = \mathrm{Diag}\left\{\frac{t_1^2\delta_1^2}{t_1^2+\delta_1^2}, \ldots, \frac{t_L^2\delta_L^2}{t_L^2+\delta_L^2}\right\}$. Next, by exploiting

the sparsity structure of $\widetilde{\mathbf{e}}_l$, $\widetilde{\mathbf{c}}_l$, and the diagonal structure of $\mathbf{T}$, we can derive a closed-form expression for the inverse of the Schur complement (i.e., $(\mathbf{I}+\mathbf{V}\mathbf{D}^{-1}\mathbf{U})^{-1}$, see [21, Theorem 4.4]). Then, through the SMW lemma, we arrive at the first key result of this paper (the proof is relegated to [21, Appendix C] due to space limitation):

Theorem 1. *$\mathbf{H}_k^{-1}$ has the same "arrow-head" structure as in $\mathbf{H}_k$, i.e.,*

$$\mathbf{H}_k^{-1}=\left[\begin{array}{c|ccc} \widehat{\mathbf{S}} & & & \\ \hline & \widehat{\mathbf{X}}_1 & & \widehat{\mathbf{C}}_1 \\ & & \ddots & \vdots \\ & & \widehat{\mathbf{X}}_L & \widehat{\mathbf{C}}_L \\ & \widehat{\mathbf{C}}_1^T & \cdots & \widehat{\mathbf{C}}_L^T \quad \widehat{\mathbf{T}} \end{array}\right], \tag{23}$$

where the closed-form expressions for $\widehat{\mathbf{S}}$, $\widehat{\mathbf{X}}_l$, $\widehat{\mathbf{C}}_l$, and $\widehat{\mathbf{T}}$ are respectively stated in [21, Eqs. (29)–(32)] due to limited space.

Thanks to the nice closed-form expressions of $\mathbf{H}_k^{-1}$, we can show that the primal Newton directions can be computed distributedly as stated in the following theorem:

Theorem 2. *In the special wireless network setting (19), given dual variables $\mathbf{w}$, the Newton direction Δs_f, $\Delta x_l^{(f)}$, Δt_l for each source rate s_f, link flow rate $x_l^{(f)}$, and link time-sharing variable t_l can be computed using local information at each source node s and link l, respectively, as follows:*

$$\Delta s_f=\frac{s_f\left(\mu s_f U_f'(s_f)+1-s_f w_{\mathrm{Src}(f)}^{(f)}\right)}{1-\mu s_f^2 U_f''(s_f)}, \tag{24}$$

$$\Delta x_l^{(f)}=\left(x_l^{(f)}\right)^2\left[\left(1-(1-R_{l,1})\frac{(x_l^{(f)})}{\|\hat{\mathbf{x}}_l\|^2}\right)\left(\frac{1}{x_l^{(f)}}-\frac{1}{\delta_l}+ w_{\mathrm{Tx}(l)}^{(f)}-w_{\mathrm{Rx}(l)}^{(f)}\right)+R_{l,1}(x_l^{(f)})^2\left(\frac{1}{t_l}-\frac{1}{\delta_l}-w\right)-\sum_{f'=1,\neq f}^{F}(1-R_{l,1})(x_l^{(f')})^2\left(\frac{1}{x_l^{(f')}}-\frac{1}{\delta_l}+w_{\mathrm{Tx}(l)}^{(f')}-w_{\mathrm{Rx}(l)}^{(f')}\right)\right], \tag{25}$$

$$\Delta t_l=R_{l,1}\left[\sum_{f=1}^{F}(x_l^{(f)})^2\left(\frac{1}{x_l^{(f)}}-\frac{1}{\delta_l}+w_{\mathrm{Tx}(l)}^{(f)}-w_{\mathrm{Rx}(l)}^{(f)}\right)\right]+R_{l,2}\left(\frac{1}{t_l}-\frac{1}{\delta_l}-w\right), \tag{26}$$

where the expressions of $R_{l,1}$ and $R_{l,2}$ are respectively stated in [21, Eqs. (26)–(27)] due to space limitation.

Sketch of the Proof: The basic idea of proving Theorem 2 is to apply Theorem 1 and exploit the second-order prosperities of the $\mathbf{a}_l^{(f)}$ and $\mathbf{b}^{(f)}$ vectors to simplify the result. We relegate the proof details to [21, Appendix D]. ∎

Remark 2. *In addition to providing closed-form expressions for primal Newton directions, Theorem 2 offers an interesting networking interpretation. Here, we can think of the difference of the dual variables $(w_{\mathrm{Tx}(l)}^{(f)}-w_{\mathrm{Rx}(l)}^{(f)})$ in (25) and (26) as "the queue length difference" in the back-pressure algorithm: if the positive $(w_{\mathrm{Tx}(l)}^{(f)}-w_{\mathrm{Rx}(l)}^{(f)})$–values outweigh the negative ones, i.e., the "pressure" on the transmitter side of link l is greater than the "pressure" on the receiver side, then $x_l^{(f)}$ and t_l will be increased in the next iteration.*

2) Primal Newton Directions: General Network Settings. So far, we have used an SMW-based approach to derive a fully distributed computational scheme for the primal Newton directions under the special network setting. For general network settings, however, the SMW-based approach usually fails because the $\mathbf{c}_l$-vectors are not orthogonal to each other due to the arbitrary interference relationships. As a result, the Schur complement $\mathbf{I}+\mathbf{V}\mathbf{D}^{-1}\mathbf{U}$ becomes a dense matrix, which yields little special structure to exploit, and makes the computation of $(\mathbf{I}+\mathbf{V}\mathbf{D}^{-1}\mathbf{U})^{-1}$ equally difficult comparing to that of $\mathbf{H}_k^{-1}$ itself. To handle this challenge, we now propose a matrix-splitting technique to compute the primal Newton directions for general network settings.

Simply speaking, matrix-splitting is a generic framework for solving linear equation systems iteratively [22]. Consider a consistent linear equation system $\mathbf{F}\mathbf{z}=\mathbf{d}$, where $\mathbf{F}$ is nonsingular. Split $\mathbf{F}$ as $\mathbf{F}=\mathbf{F}_1-\mathbf{F}_2$, where $\mathbf{F}_1$ is also nonsingular. Clearly, there are multiple choices in splitting $\mathbf{F}$. A good splitting strategy, however, is to choose an $\mathbf{F}_1$ that is easier to invert than $\mathbf{F}$. Then, it follows that $\mathbf{z}=(\mathbf{F}_1^{-1}\mathbf{F}_2)\mathbf{z}+\mathbf{F}_1^{-1}\mathbf{d}$. Now, let $\mathbf{z}^0$ be an arbitrary starting vector and consider the following iterative scheme [22]:

$$\mathbf{z}^{k+1}=(\mathbf{F}_1^{-1}\mathbf{F}_2)\mathbf{z}^k+\mathbf{F}_1^{-1}\mathbf{d},\quad k\geq 0. \tag{27}$$

It can be shown that the iterative scheme in (27) converges to the unique solution $\mathbf{z}=\mathbf{F}^{-1}\mathbf{d}$ if and only if $\rho(\mathbf{F}_1^{-1}\mathbf{F}_2)<1$, where $\rho(\cdot)$ represents the spectral radius of a matrix.

Now, we apply the matrix-splitting idea in (27) to compute $\Delta\mathbf{y}^k$. But before deriving the details of the matrix-splitting scheme, we point out that due to the block diagonal structure between the $\mathbf{S}$-block and the rest of $\mathbf{H}_k$ (cf. Eq. (18)), we are still able to compute Δs_f in *closed-form* by (24) even in general wireless network settings. As a result, the matrix-splitting scheme is only needed to compute $\Delta x_l^{(f)}$ and Δt_i. Now, we let $\overline{\mathbf{H}}_k$ denote the submatrix block obtained by removing $\mathbf{S}$ and its associated rows and columns from (18). We define the following matrices: $\mathbf{\Lambda}_k=\mathrm{Diag}\left\{\mathrm{diag}\left\{\overline{\mathbf{H}}_k\right\}\right\}$ and $\mathbf{\Omega}_k=\overline{\mathbf{H}}_k-\mathbf{\Lambda}_k$. Further, let $\overline{\mathbf{\Omega}}_k$ be a diagonal matrix with diagonal entries given by $(\overline{\mathbf{\Omega}}_k)_{ii}=\sum_j|(\mathbf{\Omega}_k)_{ij}|$. Then, we have the following result:

Lemma 3. *Let $\overline{\mathbf{H}}_k$ be split as $\overline{\mathbf{H}}_k=(\mathbf{\Lambda}_k+\alpha\overline{\mathbf{\Omega}}_k)-(\alpha\overline{\mathbf{\Omega}}_k-\mathbf{\Omega}_k)$, where $\alpha>\frac{1}{2}$ is a parameter for tuning convergence speed. Then, the sequence $\{\Delta\mathbf{y}_m^k\}_{m=1}^{\infty}$ generated by*

$$\Delta\mathbf{y}_{m+1}^k=(\mathbf{\Lambda}_k+\alpha\overline{\mathbf{\Omega}}_k)^{-1}(\alpha\overline{\mathbf{\Omega}}_k-\mathbf{\Omega}_k)\Delta\mathbf{y}_m^k+(\mathbf{\Lambda}_k+\alpha\overline{\mathbf{\Omega}}_k)^{-1}(-\nabla f_\mu(\mathbf{y}^k)-\mathbf{M}^T\mathbf{w}^k) \tag{28}$$

converges to $\Delta\mathbf{y}^k$ in (11) as $m\to\infty$.

Sketch of the proof: The key idea of proving Lemma 3 is to verify that both the sum and difference of the two components in the splitting scheme are strictly *diagonally*

dominant. Then, the result follows from Lemmas 4.7 and 4.8 in [21]. We relegate the proof details to [21, Appendix E]. ■

Remark 3. *Lemma 3 is inspired by, and is also a generalization of, the matrix-splitting scheme in [10]. In both schemes, the basic idea is to construct a diagonal nonsingular matrix for which the inverse can be distributedly computed. However, our scheme is parameterized by α, which enables convergence speed tuning in (28), while the scheme in [10] can be viewed as a special case of our scheme with $\alpha \equiv 1$.*

Remark 4. *We show in [21, Lemma 4.10] that if $\alpha_1 \leq \alpha_2$, then $\rho_{\alpha_1} \leq \rho_{\alpha_2}$. This suggests that in order for (28) to converge faster, one should choose a smaller α, provided that (28) is convergent. We point out that the technical condition $\alpha > \frac{1}{2}$ in Lemma 3 is a sufficient condition to guarantee the convergence of (28). In practical implementations, $\alpha \leq \frac{1}{2}$ could be used for faster convergence as long as $\rho((\mathbf{\Lambda}_k + \alpha\overline{\mathbf{\Omega}}_k)^{-1}(\alpha\overline{\mathbf{\Omega}}_k - \mathbf{\Omega}_k)) < 1$. This eigen-spectrum condition, however, is more inconvenient to check.*

Next, we show that the matrix-splitting scheme in Lemma 3 can be implemented in a distributed fashion to compute the primal Newton directions. We state the result as follows:

Theorem 4. *In general wireless settings, let $\Delta x_{l,m}^{(f)}$, and $\Delta t_{i,m}$ denote the the values of $\Delta x_l^{(f)}$ and Δt_i in the m-th iteration, respectively. Given dual variables $\mathbf{w}^k$, the Newton directions $\Delta x_l^{(f)}$ and Δt_i can be iteratively computed as follows:*

$$\Delta x_{l,m+1}^{(f)} = \frac{1}{P_{l,1}^{(f)}}\left[P_{l,2}^{(f)} + \left(\frac{1}{x_l^{(f)}} - \frac{1}{\delta_l} + w_{\mathrm{Tx}(l)}^{(f)} - w_{\mathrm{Rx}(l)}^{(f)}\right)\right], \quad (29)$$

$$\Delta t_{i,m+1} = \frac{1}{Q_{i,1}}\left[Q_{i,2} + \left(\frac{1}{t_i} + \sum_{l=1}^{L}\left(\frac{C_l^{(i)}}{\delta_l}\right)\right) - w\right], \quad (30)$$

where $P_{l,1}^{(f)}$, $P_{l,2}^{(f)}$, $Q_{i,1}$, and $Q_{i,2}$ are, respectively, stated in Eqs. (43)–(46) in [21] due to space limitation.

Sketch of the proof: Theorem 4 can be proved by using the "arrow-head" structure of $\overline{\mathbf{H}}_k$ and exploiting the second-order properties of $\mathbf{a}_l$ and $\widetilde{\mathbf{b}}^{(f)}$ to compute the element-wise expansion of (28). See [21, Appendix G] for more details. ■

D. Distributed Computation of the Dual Variables

Recall that the dual variables $\mathbf{w}^k$ can be computed by solving the linear equation system in (12). However, there remain two key technical challenges in dual updates:

C1) $\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T$ (i.e., the weighted Laplacian matrix) is clearly a dense matrix and it is intractable to derive any closed-form result for its inverse even in the special setting.

C2) Due to the lack of closed-form expressions for $\mathbf{H}_k^{-1}$ in the general setting, $\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T$ and $-\mathbf{M}\mathbf{H}_k^{-1}\nabla f_\mu(\mathbf{y}^k)$ cannot be written explicitly in terms of the $\mathbf{s}$-, $\mathbf{x}$-, and $\mathbf{t}$-variables. As a result, the basic matrix-splitting technique in (27) fails to compute $(\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T)^{-1}$ and $-\mathbf{M}\mathbf{H}_k^{-1}\nabla f_\mu(\mathbf{y}^k)$, let alone their distributed implementations.

In what follows, according to the different approaches to obtain $\mathbf{H}_k^{-1}$, we again classify the dual updates into special and general settings and combat the above difficulties separately.

1) Dual updates: The Special Network Setting. Dual updates in the special setting is relatively easier thanks to the closed-form result of $\mathbf{H}_k^{-1}$ in Theorem 1. By exploiting the blockwise structures in $\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T$ and $-\mathbf{M}\mathbf{H}_k^{-1}\nabla f_\mu(\mathbf{y}^k)$ (see [21, Eqs. (65)–(66)]), it can be shown that the distributed dual updates can again be done following the basic matrix-splitting scheme in (27) plus a convergence speed tuning parameter. Due to space limitation, we refer readers to [21, Theorem 4.17] for further details.

2) Dual updates: General Network Settings. As mentioned in C2), the basic matrix-splitting technique fails in the general settings. To address this challenge, we propose a new *double matrix-splitting* technique to compute $(\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T)^{-1}$ and $-\mathbf{M}\mathbf{H}_k^{-1}\nabla f_\mu(\mathbf{y}^k)$ and show that they can also be implemented in a distributed fashion. In what follows, we outline the key steps. First, consider $\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T$, for which we have the following decomposition result:

Lemma 5. $\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T$ *can be decomposed into:*

$$\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T = \widetilde{\mathbf{B}}_0\mathbf{S}^{-1}\widetilde{\mathbf{B}}_0^T + \widehat{\mathbf{M}}\overline{\mathbf{H}}_k^{-1}\widehat{\mathbf{M}}^T, \quad (31)$$

where $\widetilde{\mathbf{B}}_0 \triangleq [\widetilde{\mathbf{B}}^T, \mathbf{0}_F]^T$ and $\widehat{\mathbf{M}} \triangleq \mathrm{Diag}\{[\mathbf{A}_1, \ldots, \mathbf{A}_L], 1_I^T\}$.

Due to the (block) diagonal structure of $\widetilde{\mathbf{B}}$ and $\mathbf{S}^{-1}$, the term $\widetilde{\mathbf{B}}_0\mathbf{S}^{-1}\widetilde{\mathbf{B}}_0^T$ in (31) can be easily computed in a distributed fashion at each source node. However, the main challenge arises from $\widehat{\mathbf{M}}\overline{\mathbf{H}}_k^{-1}\widehat{\mathbf{M}}^T$, where we do not have closed-form expressions for $\overline{\mathbf{H}}_k^{-1}$. Fortunately, a closer look at $\widehat{\mathbf{M}}\overline{\mathbf{H}}_k^{-1}\widehat{\mathbf{M}}^T$ reveals that it can be further decomposed into:

$$\widehat{\mathbf{M}}\overline{\mathbf{H}}_k^{-1}\widehat{\mathbf{M}}^T = \widehat{\mathbf{M}}\mathbf{Z}_k, \quad \text{and} \quad \mathbf{Z}_k = \overline{\mathbf{H}}_k^{-1}\widehat{\mathbf{M}}^T.$$

Further, $\mathbf{Z}_k = \overline{\mathbf{H}}_k^{-1}\widehat{\mathbf{M}}^T$ can be written as $\mathbf{z}_j^{(k)} = \overline{\mathbf{H}}_k^{-1}\widehat{\mathbf{m}}_j$, $\forall j$, where $\mathbf{z}_j^{(k)}$ and $\widehat{\mathbf{m}}_j$ are the j-th columns in $\mathbf{Z}_k$ and $\widehat{\mathbf{M}}$, respectively. Now, it is important to recognize that the expression $\mathbf{z}_j^{(k)} = \overline{\mathbf{H}}_k^{-1}\widehat{\mathbf{m}}_j$ is in a similar form as in (11). Hence, $\mathbf{z}_j^{(k)}$ can be computed by the basic matrix-splitting technique similar to that in the primal Newton directions (iteration index k is omitted for simplicity):

Proposition 6 (First layer of matrix-splitting)**.** *Let $z_j^{(x_l^{(f)})}$ and $z_j^{(t_i)}$ denote the entries in $\mathbf{z}_j^{(k)}$ that correspond to variables $x_l^{(f)}$ and t_i, respectively. Let $z_{j,m}^{(x_l^{(f)})}$ and $z_{j,m}^{(t_i)}$ denote the m-th iteration values of $z_j^{(x_l^{(f)})}$ and $z_j^{(t_i)}$, respectively. Then, $\mathbf{z}_j^{(k)}$ can be iteratively computed by:*

$$z_{j,m+1}^{(x_l^{(f)})} = \frac{1}{P_{l,1}^{(f)}}\left[P_{l,3}^{(f)}(j) + P_{l,4}^{(f)}(j)\right], \qquad \forall l, f, \quad (32)$$

$$z_{j,m+1}^{(t_i)} = \frac{1}{Q_{i,1}}\left[Q_{i,3}(j) + Q_{i,4}(j)\right], \qquad \forall i, \quad (33)$$

where $P_{l,1}^{(f)}$, $Q_{i,1}$ are the same as in Theorem 4; $P_{l,3}^{(f)}(j)$, $P_{l,4}^{(f)}(j)$, $Q_{i,3}(j)$, $Q_{i,4}(j)$ are stated in [21, Eqs. (52)–(55)].

The proof of Proposition 6 is similar to that of Theorem 4, and we regelate the details to [21, Appendix H].

Remark 5. *We point out that the z-variables in (32) and (33) are obtained by using the* same *matrix-splitting scheme as in (29) and (30). Therefore, they can be computed along with the primal Newton directions in (29) and (30) by sharing $P_{l,1}^{(f)}$ and $Q_{i,1}$, thus saving a significant amount of computing resources.*

With $\mathbf{z}_j^{(k)}$, it can be shown that $\widehat{\mathbf{M}}\overline{\mathbf{H}}_k^{-1}\widehat{\mathbf{M}}^T = \widehat{\mathbf{M}}\mathbf{Z}_k$ can be computed distributedly as follows (see [21, Appendix I] for proof details):

Proposition 7. *Let $\beta_f(n)$, $n \neq \mathrm{Dst}(f)$, be an index function such that $\beta_f(n) = n$ if $n < \mathrm{Dst}(f)$ or $n-1$ if $n > \mathrm{Dst}(f)$. Let $(\widehat{\mathbf{M}}\mathbf{Z}_k)_{j_1 j_2}$ be the entry in the j_1-th row and j_2-column of $\widehat{\mathbf{M}}\mathbf{Z}_k$. Then, $(\widehat{\mathbf{M}}\mathbf{Z}_k)_{j_1 j_2}$ can be distributedly computed as*

$$(\widehat{\mathbf{M}}\mathbf{Z}_k)_{j_1 j_2} = \begin{cases} \sum_{l\in\mathcal{O}(n)} z_{j_2}^{(x_l^{(f)})} - \sum_{l\in\mathcal{I}(n)} z_{j_2}^{(x_l^{(f)})}, \\ \quad \text{if } j_1 = (f-1)(N-1)+n,\ n \neq \beta_f(\mathrm{Src}(f)), \\ \quad \text{and } j_2 = (f'-1)(N-1)+n', \\ z_{j_2}^{(s_f)} + \sum_{l\in\mathcal{O}(n)} z_{j_2}^{(x_l^{(f)})} - \sum_{l\in\mathcal{I}(n)} z_{j_2}^{(x_l^{(f)})}, \\ \quad \text{if } j_1 = (f-1)(N-1)+n,\ n \neq \beta_f(\mathrm{Src}(f)), \\ \quad \text{and } j_2 = (f'-1)(N-1)+n', \\ \sum_{i=1}^{I} z_{j_2}^{(t_i)}, \text{if } j_1 = (N-1)F+1, \\ \quad \text{and } j_2 = 1,\ldots,(N-1)F+1. \end{cases} \tag{34}$$

Next, we consider $-\mathbf{M}\mathbf{H}_k^{-1}\nabla f_\mu(\mathbf{y}^k)$, which can be decomposed into $\widetilde{\mathbf{B}}_0\mathbf{S}^{-1}(-\nabla_{\mathbf{s}} f_\mu(\mathbf{y}^k)) + \widehat{\mathbf{M}}\overline{\mathbf{H}}_k^{-1}(-\nabla_{\mathbf{x},\mathbf{t}} f_\mu(\mathbf{y}^k))$, where $\nabla_{\mathbf{s}} f_\mu(\mathbf{y}^k)$ and $\nabla_{\mathbf{x},\mathbf{t}} f_\mu(\mathbf{y}^k)$ represent the partial derivatives with respect to the $\mathbf{s}$-variables and remaining variables, respectively. Following the same approach, it can be shown that $\widetilde{\mathbf{B}}_0\mathbf{S}^{-1}(-\nabla_{\mathbf{s}} f_\mu(\mathbf{y}^k))$ can be distributedly computed at each source node. $\widehat{\mathbf{M}}\overline{\mathbf{H}}_k^{-1}(-\nabla_{\mathbf{x},\mathbf{t}} f_\mu(\mathbf{y}^k))$ can be further decomposed into:

$$\widehat{\mathbf{M}}\overline{\mathbf{H}}_k^{-1}(-\nabla_{\mathbf{x},\mathbf{t}} f_\mu(\mathbf{y}^k)) = \widehat{\mathbf{M}}\mathbf{g}, \quad \mathbf{g} = \overline{\mathbf{H}}_k^{-1}(-\nabla_{\mathbf{x},\mathbf{t}} f_\mu(\mathbf{y}^k)).$$

Then, the distributed computations of $\mathbf{g}$ and $\widehat{\mathbf{M}}\mathbf{g}$ follow from the same approach as in Propositions 6 and 7. We omit the results here for brevity and refer readers to [21, Propositions 4.14 and 4.15] for details. Finally, based on the above results, we can compute $\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T$ and $-\mathbf{M}\mathbf{H}_k^{-1}\nabla f_\mu(\mathbf{y}^k)$ without explicitly knowing $\mathbf{H}_k^{-1}$.

Now, we are ready to use the second layer of matrix-splitting to compute dual updates. To this end, we define the following matrices: $\mathbf{\Pi}_k = \mathrm{Diag}\{\mathrm{diag}\{\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T\}\}$ and $\mathbf{\Psi}_k = \mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T - \mathbf{\Pi}_k$. Also, let $\overline{\mathbf{\Psi}}$ be the diagonal matrix with diagonal entries given as $(\overline{\mathbf{\Psi}})_{ii} = \sum_j |(\overline{\mathbf{\Psi}})_{ij}|$. Then, adopting the same approach as in Lemma 3, we have

Proposition 8 (Second layer of matrix-splitting)**.** *Split $\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T$ as $\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T = (\mathbf{\Pi}_k + \alpha\overline{\mathbf{\Psi}}_k) - (\alpha\overline{\mathbf{\Psi}}_k - \mathbf{\Psi}_k)$, where $\alpha > \frac{1}{2}$ is a parameter for tuning convergence speed. Then, the sequence $\{\mathbf{w}_m^k\}_{m=1}^{\infty}$ generated by $\mathbf{w}_{m+1}^k = (\mathbf{\Pi}_k + \alpha\overline{\mathbf{\Psi}}_k)^{-1}(\alpha\overline{\mathbf{\Psi}}_k - \mathbf{\Psi}_k)\mathbf{w}_m^k + (\mathbf{\Pi}_k + \alpha\overline{\mathbf{\Psi}}_k)^{-1}(-\mathbf{M}\mathbf{H}_k^{-1}\nabla f_\mu(\mathbf{y}^k))$ converges to $\mathbf{w}^k$ in (12) as $m \to \infty$.*

Algorithm 1 Distributed Newton's Method for Problem CLO.

Initialization:

1. Each source and link: Choose some appropriate values for s_f, $x_l^{(f)}$, $\forall f$, and t_i, $\forall i$.
2. Each node: Choose appropriate values for $w_n^{(f)}$, $\forall f$ and w.

Main Iteration:

3. *Primal Newton directions (special setting)*: Update Δs_f, $\Delta x_l^{(f)}$, and Δt_l using (24), (25), and (26) at each source node and link, respectively.
4. *Primal Newton directions (general settings)*: Update Δs_f, $\Delta x_l^{(f)}$, and Δt_l using (24), (29), and (30) at each source node and link, respectively. Meanwhile, compute and store $\mathbf{z}_j^{(k)}$, $\forall j$, using (32) and (33); compute and store $\mathbf{g}$ using [21, Proposition 4.14].
5. *Dual updates (special setting)*: Update $w_n^{(f)}$ and w using [21, Theorem 4.17] at each node.
6. *Dual updates (general settings)*: First compute $\widehat{\mathbf{M}}\overline{\mathbf{H}}_k^{-1}\widehat{\mathbf{M}}^T$ using (34) and the $\mathbf{z}_j^{(k)}$-vectors obtained from Step 4. Also, compute $-\widehat{\mathbf{M}}\overline{\mathbf{H}}_k^{-1}\nabla_{\mathbf{x},\mathbf{t}} f_\mu(\mathbf{y}^k)$ using [21, Proposition 4.15] and the $\mathbf{g}$-vector obtained from Step 4. Then, update $w_n^{(f)}$ and w using Proposition 8.
7. Terminate the algorithm if a predefined run-time limit is reached or if the Newton decrement criterion is satisfied. Otherwise, go to Step 3 for the special setting case or to Step 4 for the general setting case.

So far, we have derived the main components of our distributed Newton's method. We point out that there are several topics remained to be discussed for its implementation, e.g., information exchange scale, initialization of the algorithm, stopping criterion, step-size selection, etc. We refer readers to [21, Section 4.5] for further discussions on these topics.

To conclude this section, we summarize our distributed Newton's method in Algorithm 1. In Algorithm 1, after initialization in Steps 1-2, Steps 3 and 4 are for computing primal Newton directions $\Delta\mathbf{y}^k$ in (11) under the special and general settings, respectively; while Steps 5 and 6 are for dual updates $\mathbf{w}^k$ in (12) under the special and general settings, respectively. The main iteration stops if the criterion in Step 7 is met.

V. Numerical Results

In this section, we present some numerical results to demonstrate the efficacy of our proposed distributed Newton's method. First, we examine the convergence speed of the parameterized matrix-splitting scheme in computing the primal Newton directions $\Delta\mathbf{y}^k$ and dual variables $\mathbf{w}^k$. We use an 8-node 3-session network as an example. For $\Delta\mathbf{y}^k$ and $\mathbf{w}^k$, we vary α from 0.1 to 1. In both cases, the matrix-splitting scheme is terminated when the error between the true solution of $\Delta\mathbf{y}^k$ in Eq. (11) (resp., $\mathbf{w}^k$ in Eq. (12)) and the matrix-splitting based solution is less than 1×10^{-6}. The errors for $\Delta\mathbf{y}^k$ and $\mathbf{w}^k$ are plotted (in log scale) in top and bottom halves in Fig. 1, respectively. We can see that for all values of α, the errors decrease exponentially. Also, the smaller the value of α, the faster the convergence speed. For example, when $\alpha = 0.1$, the number of iterations is approximately half of that when $\alpha = 1$ (9 vs. 15 in the primal case and 27 vs. 53 in the dual case). This confirms our analysis of the choice of α in Remark 4. To illustrate the convergence behavior of our distributed Newton's method, we use a network example as shown in Fig. 2, where five nodes are distributed in a region of $800m \times 800m$. The network is assumed to be operating under the special setting with normalized link capacity. There are two sessions in the

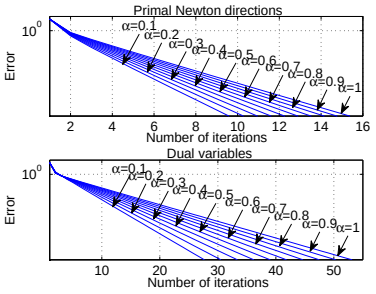

Fig. 1. Convergence behavior of the matrix-splitting scheme.

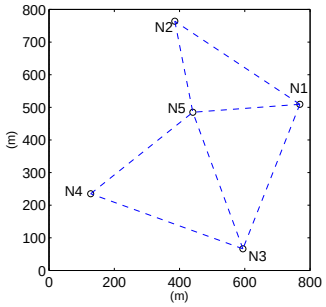

Fig. 2. A five-node two-session network.

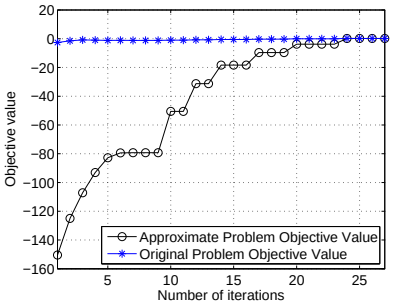

Fig. 3. Convergence behavior for the network in Fig. 2.

network: N5 to N4 and N1 to N3. We adopt $\log(s_f)$ as our utility function, which is a well-known model for "proportional fairness" [1]. The convergence behavior is illustrated in Fig. 3, which shows both the objective values of the approximating and the original problems. It can be seen that our proposed algorithm takes only 27 iterations to converge. To compare our algorithm with the subgradient method, we randomly generate 50 networks with 30 nodes and six sessions. For these 50 examples, the mean numbers of iterations for our method and the subgradient method are 720.58 and 53870.12, respectively, which shows that our proposed algorithm converges two orders of magnitude faster.

VI. Conclusion

In this paper, we developed new second-order distributed methods for cross-layer optimization in wireless networks. We first considered a special network setting where all links mutually interfere with each other. In this case, we derived *closed-form expressions* for the Hessian inverse, which further yielded a distributed implementation of the Newton's method. For general wireless networks where the interference relationships are arbitrary, we proposed a *double matrix-splitting scheme* to compute the primal Newton directions and dual variables, respectively, which also led to a distributed implementation of the Newton's method. Collectively, these results serve as the first building block of a new second-order theoretical framework for cross-layer optimization in wireless networks. Distributed second-order methods for wireless networks is an important and yet under-explored area. Future research topics may include to incorporate signal to interference plus noise ratio (SINR) based interference models, to analyze the impact of inexact line searches on convergence, to design efficient scheduling schemes, and to consider stochastic traffic models.

References

[1] X. Lin, N. B. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1452–1463, Aug. 2006.

[2] L. Tassiulas and A. Ephremides, "Stability properties of constrained queuing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[3] X. Lin and N. B. Shroff, "Joint rate control and scheduling in multihop wireless networks," in *Proc. IEEE CDC*, Atlantis, Paradise Island, Bahamas, Dec. 2006, pp. 1484–1489.

[4] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89–103, Jan. 2005.

[5] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," in *Proc. IEEE INFOCOM*, Miami, FL, Mar. 2005, pp. 1804–1814.

[6] X. Lin and N. B. Shroff, "The impact of imperfect scheduling on cross-layer congestion control in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, pp. 302–315, Apr. 2006.

[7] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd ed. New York, NY: John Wiley & Sons Inc., 2006.

[8] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

[9] A. Jadbabaie, A. Ozdaglar, and M. Zargham, "A distributed Newton method for network optimization," in *Proc. IEEE Conference on Decision and Control (CDC)*, Shanghai, China, Dec. 16-18, 2009.

[10] E. Wei, A. Ozdaglar, and A. Jadbabaie, "A distributed Newton method for network utitlity maximization," in *Proc. IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, Dec. 15-17, 2010.

[11] J. Liu and H. D. Sherali, "A distributed Newton's method for joint multi-hop routing and flow control: Theory and algorithm," in *Proc. IEEE INFOCOM*, Orlando, FL, Mar. 25-30, 2012, pp. 2489–2497.

[12] D. P. Bertsekas and E. M. Gafni, "Projected Newton methods and optimization of multi-commodity flows," *IEEE Trans. Autom. Control*, vol. 28, no. 12, pp. 1090–1096, Dec. 1983.

[13] J. G. Klincewicz, "A Newton method for convex separable network flow problems," *Networks*, vol. 13, no. 3, pp. 427–442, Mar. 1983.

[14] A. Zymnis, N. Trichakis, S. Boyd, and D. ONeill, "An interior-point method for large scale network utility maximization," in *Proc. Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Sep. 26-28, 2007.

[15] D. Bickson, Y. Tock, O. Shental, and D. Dolev, "Polynomial linear programming with Gaussian belief propagation," in *Proc. Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Sep. 23-26, 2008, pp. 895–901.

[16] D. Bickson, Y. Tock, A. Zymnis, S. Boyd, and D. Dolev, "Distributed large scale network utility maximization," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, Seoul, Korea, Jun.28–Jul.3, 2009, pp. 829–833.

[17] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*, 3rd ed. Philadelphia, PA: SIAM, 2001.

[18] D. Bickson, "Gaussian belief propagation: Theory and application," Ph.D. dissertation, Hebrew University of Jerusalem, 2009.

[19] F. R. K. Chung, *Spectral Graph Theory*. Providence, RI: American Mathematical Society, 1994.

[20] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*, 4th ed. New York: John Wiley & Sons Inc., 2010.

[21] J. Liu, C. H. Xia, N. B. Shroff, and H. D. Sherali, "Distributed cross-layer optimization in wireless networks: A second-order approach," *Technical Report, Dept. of ECE, Ohio State University*, Jul. 2012. [Online]. Available: http://www2.ece.ohio-state.edu/~liu/publications/DNewton_Wireless.pdf

[22] Z. I. Woznicki, "Matrix splitting principles," *International Journal of Mathematics and Mathematical Sciences*, vol. 28, no. 5, pp. 251–284, May 2001.