Distributed Optimization Using the Primal-Dual Method of Multipliers
Guoqiang Zhang and Richard Heusdens

Abstract—In this paper, we propose the primal-dual method of multipliers (PDMM) for distributed optimization over a graph. In particular, we optimize a sum of convex functions defined over a graph, where every edge in the graph carries a linear equality constraint. In designing the new algorithm, an augmented primal-dual Lagrangian function is constructed which smoothly captures the graph topology. It is shown that a saddle point of the constructed function provides an optimal solution of the original problem. Further under both the synchronous and asynchronous updating schemes, PDMM has the convergence rate of $O(1/K)$ (where $K$ denotes the iteration index) for general closed, proper and convex functions. Other properties of PDMM such as convergence speeds versus different parameter-settings and resilience to transmission failure are also investigated through the experiments of distributed averaging.

Index Terms—Distributed optimization, ADMM, PDMM, sub-linear convergence.

I. INTRODUCTION

In recent years, distributed optimization has drawn increasing attention due to the demand for big-data processing and easy access to ubiquitous computing units (e.g., a computer, a mobile phone or a sensor equipped with a CPU). The basic idea is to have a set of computing units collaborate with each other in a distributed way to complete a complex task. Popular applications include telecommunication [3], [4], wireless sensor networks [5], cloud computing and machine learning [6]. The research challenge is on the design of efficient and robust distributed optimization algorithms for those applications.

To the best of our knowledge, almost all the optimization problems in those applications can be formulated as optimization over a graph model $G = (V, E)$:

$$\min_{\{x_i\}i \in V} \sum_{i \in V} f_i(x_i) + \sum_{(i,j) \in E} f_{ij}(x_i, x_j),$$

(1)

where $\{f_i\}i \in V$ and $\{f_{ij}\}(i,j) \in E$ are referred to as node and edge-functions, respectively. For instance, for the application of distributed quadratic optimization, all the node and edge-functions are in the form of scalar quadratic functions (see [7], [8], [9]).

G. Zhang is with the school of Computing and Communications, University of Technology, Sydney, Australia. Email: guoqiang.zhang@uts.edu.au
R. Heusdens is with the Department of Microelectronics, Circuits and Systems group, Delft University of Technology, The Netherlands. Email: r.heusdens@tudelft.nl
Part of the work has been published on ICASSP, 2015, with the paper titled Bi-Alternating Direction Method of Multipliers over Graphs. After careful consideration, we decide to change the name of our algorithm from bi Alternating direction method of multipliers (BiADMM) in [1] and [2] to primal-dual method of multipliers (PDMM).

In the literature, a large number of applications (see [10]) require that every edge function $f_{ij}(x_i, x_j), (i,j) \in E$, is essentially a linear equality constraint in terms of $x_i$ and $x_j$. Mathematically, we use $A_{ij}x_i + A_{ji}x_j = c_{ij}$ to formulate the equality constraint for each $(i,j) \in E$, as demonstrated in Fig. 1. In this situation, (1) can be described as

$$\min_{\{x_i\}i \in V} \sum_{i \in V} f_i(x_i) + \sum_{(i,j) \in E} I_{A_{ij}x_i + A_{ji}x_j = c_{ij}}(x_i, x_j),$$

(2)

where $I_{(\cdot)}$ denotes the indicator or characteristic function defined as $I_C(x) = 0$ if $x \in C$ and $I_C(x) = \infty$ if $x \notin C$. In this paper, we focus on convex optimization of form (2), where every node-function $f_i$ is closed, proper and convex.

The majority of recent research have been focusing on a specialized form of the convex problem (2), where every edge-function $f_{ij}$ reduces to $I_{x_i=x_j}(x_i, x_j)$. The above problem is commonly known as the consensus problem in the literature. Classic methods include the dual-averaging algorithm [11], the subgradient algorithm [12], the diffusion adaptation algorithm [13]. For the special case that $\{f_i\}i \in V$ are scalar quadratic functions (referred to as the distributed averaging problem), the most popular methods are the randomized gossip algorithm [5] and the broadcast algorithm [14]. See [15] for an overview of the literature for solving the distributed averaging problem.

The alternating-direction method of multipliers (ADMM) can be applied to solve the general convex optimization (2). The key step is to decompose each equality constraint $A_{ij}x_i + A_{ji}x_j = c_{ij}$ into two constraints such as $A_{ij}x_i + z_{ij} = c_{ij}$ and $z_{ij} = A_{ji}x_j$ with the help of the auxiliary variable $z_{ij}$. As a result, (2) can be reformulated as

$$\min_{x, z} f(x) + g(z) \quad \text{subject to} \quad Ax + Bz = c,$$

(3)

where $f(x) = \sum_{i \in V} f_i(x_i)$, $g(z) = 0$ and $z$ is a vector obtained by stacking up $z_{ij}$ one after another. See [16]...
for using ADMM to solve the consensus problem of (2) (with edge-function $I_{s_{i;j}}(x_i, x_j)$). The graphic structure is implicitly embedded in the two matrices $(A, B)$ and the vector $c$. The reformulation essentially converts the problem on a general graph with many nodes (2) to a graph with only two nodes (3), allowing the application of ADMM. Based on (3), ADMM then constructs and optimizes an augmented Lagrangian function iteratively with respect to $(x, z)$ and a set of Lagrangian multipliers. We refer to the above procedure as synchronous ADMM as it updates all the variables at each iteration. Recently, the work of [17] proposed asynchronous ADMM, which optimizes the same function over a subset of the variables at each iteration.

We note that besides solving (2), ADMM has found many successful applications in the fields of signal processing and machine learning (see [10] for an overview). For instance, in [18] and [19], variants of ADMM have been proposed to solve a (possibly nonconvex) optimization problem defined over a graph with a star topology, which is motivated from big data applications. The work of [20] considers solving the consensus problem of (2) (with edge-function $I_{s_{i;j}}(x_i, x_j)$) over a general graph, where each node function $f_i$ is further expressed as a sum of two component functions. The authors of [20] propose a new algorithm which includes ADMM as a special case when one component function is zero. In general, ADMM and its variants are quite simple and often provide satisfactory results after a reasonable number of iterations, making it a popular algorithm in recent years.

In this paper, we tackle the convex problem (2) directly instead of relying on the reformulation (3). Specifically, we construct an augmented primal-dual Lagrangian function for (2) without introducing the auxiliary variable $z$ as is required by ADMM. We show that solving (2) is equivalent to searching for a saddle point of the augmented primal-dual Lagrangian. We then propose the primal-dual method of multipliers (PDMM) to iteratively approach one saddle point of the constructed function. It is shown that for both the synchronous and asynchronous updating schemes, the PDMM converges with the rate of $O(1/K)$ for general closed, proper and convex functions.

Further we evaluate PDMM through the experiments of distributed averaging. Firstly, it is found that the parameters of PDMM should be selected by a rule (see VI-C1) for fast convergence. Secondly, when there are transmission failures in the graph, transmission losses only slow down the convergence speed of PDMM. Finally, experimental comparison suggests that PDMM outperforms ADMM and the two gossip algorithms in [5] and [14].

This work is mainly devoted to the theoretical analysis of PDMM. In the literature, PDMM has already been successfully applied for solving a few other problems. The work of [21] investigates the efficiency of ADMM and PDMM for distributed dictionary learning. In [22], we have used both ADMM and PDMM for training a support vector machine (SVM). In the above examples it is found that PDMM outperforms ADMM in terms of convergence rate. In [23], the authors describe an application of the linearly constrained minimum variance (LCMV) beamformer for use in acoustic wireless sensor networks. The proposed algorithm computes the optimal beamformer output at each node in the network without the need for sharing raw data within the network. PDMM has been successfully applied to perform distributed beamforming. This suggests that PDMM is not only theoretically interesting but also might be powerful in real applications.

II. Problem Setting

In this section, we first introduce basic notations needed in the rest of the paper. We then make a proper assumption about the existence of optimal solutions of the problem. Finally, we derive the dual problem to (2) and its Lagrangian function, which will be used for constructing the augmented primal-dual Lagrangian function in Section III.

A. Notations and functional properties

We first introduce notations for a graphic model. We denote a graph as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, m\}$ represents the set of nodes and $\mathcal{E} = \{(i, j) | i, j \in \mathcal{V}\}$ represents the set of edges in the graph, respectively. We use $\bar{\mathcal{E}}$ to denote the set of all directed edges. Therefore, $|\bar{\mathcal{E}}| = 2|\mathcal{E}|$. The directed edge $(i, j)$ starts from node $i$ and ends with node $j$. We use $\mathcal{N}_i$ to denote the set of all neighboring nodes of node $i$, i.e., $\mathcal{N}_i = \{j | (i, j) \in \mathcal{E}\}$. Given a graph $G = (\mathcal{V}, \mathcal{E})$, only neighboring nodes are allowed to communicate with each other directly.

Next we introduce notations for mathematical description in the remainder of the paper. We use bold small letters to denote vectors and bold capital letters to denote matrices. The notation $M \succeq 0$ (or $M \succ 0$) represents a symmetric positive semi-definite matrix (or a symmetric positive definite matrix). The superscript $(\cdot)^T$ represents the transpose operator. Given a vector $y$, we use $\|y\|$ to denote its $L_2$ norm.

Finally, we introduce the conjugate function. Suppose $h : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a closed, proper and convex function. Then the conjugate of $h(\cdot)$ is defined as [24, Definition 2.1.20]

$$h^*(\delta) \doteq \max_y \delta^T y - h(y),$$

(4)

where the conjugate function $h^*$ is again a closed, proper and convex function. Let $y^*$ be the optimal solution for a particular $\delta^*$ in (4). We then have

$$\delta^* \in \partial_y h(y^*),$$

(5)

where $\partial_y h(y^*)$ represents the set of all subgradients of $h(\cdot)$ at $y^*$ (see [24, Definition 2.1.23]). As a consequence, since $h^{**} = h$, we have

$$h(y^*) = y^T \delta^* - h^*(\delta^*) = \max_{\delta} y^T \delta - h^*(\delta),$$

(6)

and we conclude that $y^* \in \partial h^*(\delta^*)$ as well.

B. Problem assumption

With the notation $G = (\mathcal{V}, \mathcal{E})$ for a graph, we first reformulate the convex problem (2) as

$$\min_x \sum_{i \in \mathcal{V}} f_i(x_i) \text{ s. t. } A_{ij}x_i + A_{ji}x_j = c_{ij} \forall (i, j) \in \mathcal{E},$$

(7)

where each function $f_i : \mathbb{R}^n_i \to \mathbb{R} \cup \{+\infty\}$ is assumed to be closed, proper and convex, and $x = [x_1^T, x_2^T, \ldots, x_m^T]^T$. 

2373-778X (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
variables \(\delta\) over (\(f\)). Indeed, every \(L(4)\), satisfying Fenchel’s inequality (12), must hold with equality \(\delta_{ij}\) is obtained by vertically concatenating all \(\lambda_{ij}^\star\), \(j \in \mathcal{N}_i\), and \(A_{ij}^T\) is obtained by horizontally concatenating all \(A_{ij}\), \(j \in \mathcal{N}_i\). To clarify, the product \(A_{ij}^T\lambda_i\) in (13) equals \(A_{ij}^T\lambda_i = \sum_{j \in \mathcal{N}_i} A_{ij}^T\lambda_{ij}\).

Consequently, we let \(\lambda = [\lambda_{1}^T, \lambda_{1}^T, \ldots, \lambda_{|\mathcal{N}_i|}^T]^{T}\). In the above reformulation (13), each conjugate function \(f_i(\cdot)\) only involves the node variable \(\lambda_i\), facilitating distributed optimization.

Next we tackle the equality constraints in (13). To do so, we construct a (dual) Lagrangian function for the dual problem (13), which is given by

\[
L_d(\lambda, x, y) = \sum_{i \in \mathcal{V}} f_i^\star(A_i^T\lambda_i) + \sum_{(i,j) \in \mathcal{E}} \delta_{ij}^T c_{ij} + \sum_{(i,j) \in \mathcal{E}} \left[ y_{ij}^T(\delta_{ij} - \lambda_{ij}^\star) + y_{ij}^T(\delta_{ij} - \lambda_{ij}) \right],
\]

where \(y \) is obtained by concatenating all the Lagrangian multipliers \(y_{ij}, [i,j] \in \mathcal{E}\), one after another.

We now argue that each Lagrangian multiplier \(y_{ij}, [i,j] \in \mathcal{E}\), in (15) can be replaced by an affine function of \(x\). Suppose \((x^\star, \lambda^\star)\) is a saddle point of \(L_p\). By letting \(\lambda_{ij}^\star = \delta_{ij}^\star\) for every \([i,j] \in \mathcal{E}\), Fenchel’s inequality (12) must hold with equality at \((x^\star, \lambda^\star)\) from which we derive that

\[
0 \in \partial \lambda_{ij} \left[ f_i(\lambda_i^T A_i^T x^\star) - A_{ij} x^\star \right] = \partial \lambda_{ij} \left[ f_i(\lambda_i^T A_i^T x^\star) \right] + A_{ij} x^\star - c_{ij} \quad \forall [i,j] \in \mathcal{E}.
\]

One can then show that \((\delta^\star, \lambda^\star, y^\star)\) where \(y_{ij}^\star = A_{ij} x^\star - c_{ij}\) for every \([i,j] \in \mathcal{E}\), is a saddle point of \(L_d\). We therefore restrict the Lagrangian multiplier \(y_{ij}\) to be of the form \(y_{ij} = A_{ij} x_i - c_{ij}\), so that the dual Lagrangian becomes

\[
L_d(\delta, \lambda, x) = \sum_{i \in \mathcal{V}} \left( - f_i^\star(A_i^T \lambda_i) - \sum_{j \in \mathcal{N}_i} \lambda_{ij}^T(A_{ij} x_i - c_{ij}) \right) - \sum_{(i,j) \in \mathcal{E}} \delta_{ij}^T(c_{ij} - A_{ij} x_i - A_{ji} x_j).
\]

We summarize the result in a lemma below:

**Lemma 1.** If \((x^\star, \lambda^\star)\) is a saddle point of \(L_p(x, \delta)\), then \((\delta^\star, \lambda^\star, x^\star)\) is a saddle point of \(L_d(\delta, \lambda, x)\), where \(\lambda_{ij}^\star = \delta_{ij}^\star\) for every \([i,j] \in \mathcal{E}\).

We note that \(L_d(\delta, \lambda, x)\) might not be equivalent to \(L_d(\delta, \lambda, x)\). By inspection of the optimality conditions of (16), not every saddle point \((\delta^\star, \lambda^\star, x^\star)\) of \(L_d\) might lead to \(\lambda_{ij}^\star = \delta_{ij}^\star\) for every \([i,j] \in \mathcal{E}\).
penalty functions w.r.t. \( \lambda \) to implicitly enforce the equality constraints \( \{ x_{ij}^* = \lambda_{ji}^*, (i, j) \in E \} \).

To briefly summarize, one can alternatively solve the dual problem (13) instead of the primal problem. Further, by replacing \( y \) with an affine function of \( x \) in (15), the dual Lagrangian \( L_d(\delta, \lambda, x) \) share two variables \( x \) and \( \delta \) with the primal Lagrangian \( L_p(x, \delta) \). We will show in next section that the special form of \( L_d \) in (16) plays a crucial role for constructing the augmented primal-dual Lagrangian.

### III. Augmented Primal-Dual Lagrangian

In this section, we first build and investigate a primal-dual Lagrangian from \( L_p \) and \( L_d \). We show that a saddle point of the primal-dual Lagrangian does not always lead to an optimal solution of the primal or the dual problem.

To address the above issue, we then construct an augmented primal-dual Lagrangian by introducing two additional penalty functions. We show that any saddle point of the augmented primal-dual Lagrangian leads to an optimal solution of the primal and the dual problem, respectively.

#### A. Primal-Dual Lagrangian

By inspection of (8) and (16), we see that in both \( L_p \) and \( L_d \), the edge variables \( \delta_{ij} \) are related to the terms \( c_{ij} - A_{ij} x_i - A_{ji} x_j \). As a consequence, if we add the primal and dual Lagrangians, \( \delta_{ij} \) will cancel out and the resulting function contains node variables \( x \) and \( \lambda \) only.

We hereby define the new function as the primal-dual Lagrangian below:

**Definition 1.** The primal-dual Lagrangian is defined as

\[
L_{pd}(x, \lambda) = L_p(x, \delta) + L_d(\delta, \lambda, x) = \sum_{i \in V} \left[ f_i(x_i) - \sum_{j \in N_i} \lambda_{ji}^*(A_{ij} x_i - c_{ij}) - f_i^*(A_i^T \lambda_i) \right].
\]

\(L_{pd}(x, \lambda)\) is convex in \( x \) for fixed \( \lambda \) and concave in \( \lambda \) for fixed \( x \), suggesting that it is essentially a saddle-point problem (see [25], [26] for solving different saddle point problems). For each edge \((i, j) \in E\), the node variables \( \lambda_{ij} \) and \( \lambda_{ji} \) substitute the role of the edge variable \( \delta_{ij} \). The removal of \( \delta_{ij} \) enables to design a distributed algorithm that only involves node-oriented optimization (see next section for PDMM).

Next we study the properties of saddle points of \( L_{pd}(x, \lambda) \):

**Lemma 2.** If \( x^* \) solves the primal problem (7), then there exists a \( \lambda^* \) such that \((x^*, \lambda^*)\) is a saddle point of \( L_{pd}(x, \lambda) \).

**Proof.** If \( x^* \) solves the primal problem (7), then there exists a \( \delta^* \) such that \((x^*, \delta^*)\) is a saddle point of \( L_p(x, \delta) \) and by Lemma 1, there exist \( \lambda_{ij}^* = \delta_{ji}^* \) for every \([i, j] \in E\) so that \((\delta^*, \lambda^*, x^*)\) is a saddle point of \( L_d(\delta, \lambda, x) \).

Hence

\[
L_{pd}(x^*, \lambda^*) = L_p(x^*, \delta^*) + L_d(\delta^*, \lambda^*, x^*) \\
\leq L_p(x^*, \delta^*) + L_d(\delta^*, \lambda^*, x^*) \\
= L_{pd}(x^*, \lambda^*).
\]

The fact that \((x^*, \lambda^*)\) is a saddle point of \( L_{pd}(x, \lambda) \), however, is not sufficient for showing \( x^* \) (or \( \lambda^* \)) being optimal for solving the primal problem (7) (for solving the dual problem (13)).

**Example 1** \((x^* \text{ not optimal})\). Consider the following problem

\[
\min_{x_1, x_2} f_1(x_1) + f_2(x_2) \text{ s.t. } x_1 - x_2 = 0,
\]

where

\[
f_1(x_1) = \begin{cases} x_1 - 1 & \text{if } x_1 \geq 1 \\ 0 & \text{otherwise} \end{cases}
\]

With this, the primal Lagrangian is given by

\[
f_1^*(\delta_{12}) = f_1(x_1) + f_2(x_2) + \delta_{12}(x_1 - x_2), \text{ so that the dual function is given by } -f_1^*(\delta_{12}) - f_2^*(-\delta_{12}), \text{ where}
\]

\[
f_1^*(\delta_{12}) = f_2^*(-\delta_{12}) = \begin{cases} \delta_{12} & 0 \leq \delta_{12} \leq 1 \\ +\infty & \text{otherwise} \end{cases}
\]

Hence, the optimal solution for the primal and dual problem is \(x_1^* = x_2^* \in [-1, 1]\) and \(\delta_{12}^* = 0\), respectively. The primal-dual Lagrangian in this case is given by

\[
L_{pd}(x, \lambda) = f_1(x_1) + f_2(x_2) - f_1^*(\lambda_{1|2}) - f_2^*(-\lambda_{2|1}) \\
- x_1\lambda_{2|1} + x_2\lambda_{1|2}.
\]

One can show that every point \((x_1', x_2', \lambda_{1|2}', \lambda_{2|1}') \in \{(x_1, x_2, 0, 0) | 1 \leq x_1, x_2 \leq 1\}\) is a saddle point of \(L_{pd}(x, \lambda)\), which does not necessarily lead to \(x_1' = x_2'\).

It is clear from Example 1 that finding a saddle point of \(L_{pd}\) does not necessarily solve the primal problem (7). Similarly, one can also build another example illustrating that a saddle point of \(L_{pd}\) does not necessarily solve the dual problem (13).

#### B. Augmented primal-dual Lagrangian

The problem that not every saddle point of \(L_{pd}(x, \lambda)\) leads to an optimal point of the primal or dual problem can be solved by adding two quadratic penalty terms to \(L_{pd}(x, \lambda)\) as

\[
L_P(x, \lambda) = L_{pd}(x, \lambda) + h_p(x) - h_p(\lambda),
\]

where \(h_p(x)\) and \(h_p(\lambda)\) are defined as

\[
h_p(x) = \sum_{(i, j) \in E} \frac{1}{2} \| A_{ij} x_i + A_{ji} x_j - c_{ij} \|^2_{p_{ij}},
\]

\[
h_p(\lambda) = \sum_{(i, j) \in E} \frac{1}{2} \| \lambda_{ij} - \lambda_{ji} \|^2_{p_{ij}},
\]

where \(P = P_p \cup P_d\) and

\[
P_p = \{P_{p_{ij}} | P_{p_{ij}} > 0 \forall (i, j) \in E\}
\]

\[
P_d = \{P_{d_{ij}} | P_{d_{ij}} > 0 \forall (i, j) \in E\}.
\]

The \(2|E|\) positive definite matrices in \(P\) remain to be specified.

Let \(X = \{x | A_{ij} x_i + A_{ji} x_j = c_{ij}, \forall (i, j) \in E\}\) and \(\Lambda = \{\lambda | \lambda_{ij} = \lambda_{ji}, \forall (i, j) \in E\}\) denote the primal and dual feasible set, respectively. It is clear that \(h_p(\lambda) \geq 0\) (or \(-h_p(\lambda) \leq 0\)) with equality if and only if \((x, \lambda) \in X\) (or \(\lambda \in \Lambda\)). The introduction of the two penalty functions essentially prevents non-feasible \(x\) and/or \(\lambda\) to correspond to saddle points of \(L_P(x, \lambda)\). As a consequence, we have a saddle point theorem for \(L_P\) which states that \(x^*\) solves
the primal problem (7) if and only if there exists $\lambda^*$ such that $(x^*, \lambda^*)$ is a saddle point of $L_P(x, \lambda)$. To prove this result, we need the following lemma.

**Lemma 3.** Let $(x^*, \lambda^*)$ and $(x', \lambda')$ be two saddle points of $L_P(x, \lambda)$. Then

$$L_P(x', \lambda') = L_P(x', \lambda^*) = L_P(x^*, \lambda') = L_P(x^*, \lambda^*). \quad (23)$$

Further, $(x', \lambda^*)$ and $(x^*, \lambda')$ are two saddle points of $L_P(x, \lambda)$ as well.

**Proof.** Since $(x^*, \lambda^*)$ and $(x', \lambda')$ are two saddle points of $L_P(x, \lambda)$, we have

$$L_P(x', \lambda') \leq L_P(x', \lambda^*) \leq L_P(x^*, \lambda^*) \leq L_P(x^*, \lambda') \leq L_P(x^*, \lambda^*).$$

Combining the above two inequality chains produces (23). In order to show that $(x', \lambda^*)$ is a saddle point, we have $L_P(x', \lambda^*) \leq L_P(x', \lambda^*) = L_P(x^*, \lambda^*) \leq L_P(x^*, \lambda^*)$. The proof for $(x^*, \lambda')$ is similar.

We are ready to prove the saddle point theorem for $L_P(x, \lambda)$.

**Theorem 1.** If $x^*$ solves the primal problem (7), there exists $\lambda^*$ such that $(x^*, \lambda^*)$ is a saddle point of $L_P(x, \lambda)$. Conversely, if $(x^*, \lambda^*)$ is a saddle point of $L_P(x, \lambda)$, then $x^*$ solves the primal and the dual problem, respectively. Or equivalently, the following optimality conditions hold

$$\sum_{j \in N_i} A_{ij}^T x_j = 0 \quad \forall i \in \mathcal{V}, \quad (24)$$

$$A_{ij} x_i + A_{ij} x_j - c_{ij} = 0 \quad \forall (i, j) \in \mathcal{E}, \quad (25)$$

$$\lambda_j - \lambda_{ij} = 0 \quad \forall (i, j) \in \mathcal{E}. \quad (26)$$

**Proof.** If $x^*$ solves the primal problem, then there exists a $\lambda^*$ such that $(x^*, \lambda^*)$ is a saddle point of $L_{pd}$ by Lemma 2. Since $x^* \in X$ and $\lambda^* \in \Lambda$, we have $h_{pd}(x^*) - h_{pd}(\lambda^*) = 0$ and $\partial h_{pd}(x^*) = 0$ and $\partial \lambda h_{pd}(\lambda^*) = 0$, from which we conclude that $(x^*, \lambda^*)$ is a saddle point of $L_P(x, \lambda)$ as well.

Conversely, let $(x^*, \lambda^*)$ be a saddle point of $L_P(x, \lambda)$. We first show that $x^*$ solves the primal problem. We have from Lemma 3 that $L_P(x^*, \lambda^*) = L_P(x^*, \lambda^*)$, which can be simplified as

$$L_P(x^*, \lambda^*) = L_P(x^*, \lambda^*) + L_D(\delta^*, \lambda^*, x^*),$$

from which we conclude that $h_{pd}(x^*) = L_P(x^*, \delta^*) \leq 0$ and thus $h_{pd}(x^*) = 0$ so that $x^* \in X$. In addition, since $(x^*, \lambda^*)$ is a saddle point of $L_P(x, \lambda)$ by Lemma 3, we have

$$\sum_{j \in N_i} A_{ij}^T \delta_{ij} = \sum_{j \in N_i} A_{ij}^T \lambda_{ij} \in \partial x_i, f_i(x_i') \forall i \in \mathcal{V},$$

and we conclude that $x^*$ solves the primal problem as required. Similarly, one can show that $\lambda^*$ solves the dual problem.

Based on the above analysis, we conclude that the optimality conditions for $(x^*, \lambda^*)$ being a saddle point of $L_P$ are given by (24)-(26). The set of optimality conditions $\{c_{ij} - A_{ij} x_j \in \partial x_i, f_i(x_i') \forall i \in \mathcal{V}, A_{ij}^T \delta_{ij} = \sum_{j \in N_i} A_{ij}^T \lambda_{ij} \in \partial x_i, f_i(x_i') \forall i \in \mathcal{V},\}$ is redundant and can be derived from (24)-(26) (see (4)-(6) for the argument).

**Theorem 2.** Theorem 1 states that instead of solving the primal problem (7) or the dual problem (13), one can alternatively search for a saddle point of $L_P(x, \lambda)$. To briefly summarize, we consider solving the following min-max problem in the rest of the paper

$$(x^*, \lambda^*) = \arg \min_{x} \max_{\lambda} L_P(x, \lambda). \quad (27)$$

We will explain in next section how to iteratively approach the saddle point $(x^*, \lambda^*)$ in a distributed manner.

**IV. Primal-Dual Method of Multipliers**

In this section, we present a new algorithm named primal-dual method of multipliers (PDMM) to iteratively approach a saddle point of $L_P(x, \lambda)$. We propose both the synchronous and asynchronous PDMM for solving the problem.

**A. Synchronous updating scheme**

The synchronous updating scheme refers to the operation that at each iteration, all the variables over the graph update their estimates by using the most recent estimates from their neighbors from last iteration. Suppose $(\hat{x}^k, \hat{\lambda}^k)$ is the estimate obtained from the $k-1$th iteration, where $k \geq 1$. We compute the new estimate $(\hat{x}^{k+1}, \hat{\lambda}^{k+1})$ at iteration $k$ as

$$(\hat{x}^{k+1}, \hat{\lambda}^{k+1}) = \arg \min_{x, \lambda} \max_{\pi} L_P([\ldots, x_{i-1}, \hat{x}_i, x_{i+1}, \ldots]^T, [\ldots, \hat{x}_i^{k-1}, x_i^{k+1}, \lambda_i^{k-1}, \lambda_i^{k+1}, \ldots]^T) \quad i \in \mathcal{V}. \quad (28)$$

By inserting the expression (20) for $L_P(x, \lambda)$ into (28), the updating expression can be further simplified as

$$\hat{x}^{k+1}_i = \arg \min_{\hat{x}_i} \left[ \sum_{j \in N_i} \frac{1}{2} \| A_{ij} x_i + A_{ij} \hat{x}_j - c_{ij} \|_2^2 \right]_{\pi_{p,ij}}$$

$$- x_i^T \left( \sum_{j \in N_i} A_{ij}^T \lambda_{ij} \right) + f_i(x_i) \quad i \in \mathcal{V}. \quad (29)$$

$$\hat{\lambda}^{k+1}_i = \arg \min_{\hat{\lambda}_i} \left[ \sum_{j \in N_i} \frac{1}{2} \| \hat{\lambda}_{ij} - \hat{\lambda}_{ij}^k \|_2^2 + \lambda_i^T A_{ij} \hat{x}_j \right]_{\pi_{d,ij}}$$

$$- \lambda_i^T c_i \quad i \in \mathcal{V}. \quad (30)$$

Eq. (29)-(30) suggest that at iteration $k$, every node $i$ performs parameter-updating independently once the estimates $\{\hat{x}^k_j, \hat{\lambda}^k_{ij} \mid j \in N_i\}$ of its neighboring variables are available. In addition, the computation of $\hat{x}^{k+1}_i$ and $\hat{\lambda}^{k+1}_i$ can be carried out in parallel since $x_i$ and $\lambda_i$ are not directly related in $L_P(x, \lambda)$. We refer to (29)-(30) as node-oriented computation.

In order to run PDMM over the graph, each iteration should consist of two steps. Firstly, every node $i$ computes $(\hat{x}_i, \hat{\lambda}_i)$ by following (29)-(30), accounting for information-fusion. Second, every node $i$ sends $(\hat{x}_i, \hat{\lambda}_{ij})$ to its neighboring node $j$ for all neighbors, accounting for information-spread. We take $\hat{x}_i$ as the common message to all neighbors of node $i$ and $\hat{\lambda}_{ij}$ as a node-specific message only to neighbor $j$. In
some applications, it may be preferable to exploit broadcast transmission rather than point-to-point transmission in order to save energy. We will explain in Subsection IV-C that the transmission of \( \lambda_{ij}, j \in \mathcal{N}_i \), can be replaced by broadcast transmission of an intermediate quantity.

Finally, we consider terminating the iterates (29)-(30). One can check if the estimate \( \hat{x}_i, \hat{\lambda}_i \) becomes stable over consecutive iterates (see Corollary 1 for theoretical support).

### B. Asynchronous updating scheme

The asynchronous updating scheme refers to the operation that at each iteration, only the variables associated with one node in the graph update their estimates while all other variables keep their estimates fixed. Suppose node \( i \) is selected at iteration \( k \). We then compute \((\hat{x}^{k+1}_i, \hat{\lambda}^{k+1}_i)\) by optimizing \( L_P \) based on the most recent estimates \( \{\hat{x}^j, \hat{\lambda}^j_{|j|} | j \in \mathcal{N}_i\} \) from its neighboring nodes. At the same time, the estimates \((\hat{x}^j, \hat{\lambda}^j)\), \( j \neq i \), remain the same. By following the above computational instruction, \((\hat{x}^{k+1}_i, \hat{\lambda}^{k+1}_i)\) can be obtained as

\[
(\hat{x}^{k+1}_i, \hat{\lambda}^{k+1}_i) = \arg \min_{x_i, \lambda_i} \max_{L_P} \left( \ldots, \hat{x}^{k+1}_{i-1}, x_i^T, \hat{x}_i^{k+1} T, \ldots \right),
\]

\[
(\hat{x}^{k+1}_j, \hat{\lambda}^{k+1}_j) = (\hat{x}^k_j, \hat{\lambda}^k_j), \quad j \in \mathcal{V}, j \neq i. \tag{31}
\]

Similarly to (29)-(30), \( \hat{x}^{k+1}_i \) and \( \hat{\lambda}^{k+1}_i \) can also be computed separately in (31). Once the update at node \( i \) is complete, the node sends the common message \( \hat{x}^{k+1}_i \) and node-specific messages \( \{\hat{\lambda}^{k+1}_{ij}, j \in \mathcal{N}_i\} \) to its neighbors. We will explain in next subsection how to exploit broadcast transmission to replace point-to-point transmission.

In practice, the nodes in the graph can either be randomly activated or follow a predefined order for asynchronous parameter-updating. One scheme for realizing random node-activation is that after a node finishes parameter-updating, it randomly activates one of its neighbors for next iteration. Another scheme is to introduce a clock at each node which ticks at the times of a (random) Poisson process (see [5] for detailed information). Each node is activated only when its clock ticks. As for node-activation in a predefined order, cyclic updating scheme is most straightforward. Once node \( i \) finishes parameter-updating, it informs node \( i + 1 \) for next iteration. For the case that node \( i \) and \( i + 1 \) are not neighbors, the path from node \( i \) to \( i + 1 \) can be pre-stored at node \( i \) to facilitate the process. In Subsection V-D, we provide convergence analysis only for the cyclic updating scheme. We leave the analysis for other asynchronous schemes for future investigation.

**Remark 1.** To briefly summarize, synchronous PDMM scheme allows faster information-spread over the graph through parallel parameter-updating while asynchronous PDMM scheme requires less effort from node-coordination in the graph. In practice, the scheme-selection should depend on the graph (e.g., wireless sensor networks) properties such as the feasibility of parallel computation, the complexity of node-coordination and the life time of nodes.

### C. Simplifying node-based computations and transmissions

It is clear that for both the synchronous and asynchronous schemes, each activated node \( i \) has to perform two minimizations: one for \( \hat{x}_i \) and the other one for \( \hat{\lambda}_i \). In this subsection, we show that the computations for the two minimizations can be simplified. We will also study how the point-to-point transmission can be replaced with broadcast transmission. To do so, we will consider two scenarios:

1) **Avoiding conjugate functions:** In the first scenario, we consider using \( f_1(\cdot) \) instead of \( f^*_1(\cdot) \) to update \( \hat{\lambda}_i \). Our goal is to simplify computations by avoiding the derivation of \( f^*_1(\cdot) \).

By using the definition of \( f_1^* \) in (4), the computation (30) for \( \lambda_{ij}^{k+1} \) (which also holds for asynchronous PDMM) can be rewritten as

\[
\hat{\lambda}_{ij}^{k+1} = \arg \min_{\lambda_{ij}} \left[ \sum_{j \in \mathcal{N}_i} \frac{1}{2} \| \lambda_{ij} - \hat{\lambda}_{ij}^{k} \|^2_{P_{d,ij}} + \lambda_{ij} A_{ij} \hat{x}_j + \lambda_{ij} c_{ij} \right] + \max_{u_i} \left( w_i^T A_i^T \lambda_i - f_i(w_i) \right). \tag{33}
\]

We denote the optimal solution for \( w_i \) in (33) as \( w_i^{k+1} \). The optimality conditions for \( \hat{\lambda}_{ij}^{k+1}, j \in \mathcal{N}_i \), and \( w_i^{k+1} \) can then be derived from (33) as

\[
0 \in A_i^T \hat{\lambda}_{ij}^{k+1} - \partial_{w_i} f_i(w_i^{k+1}) \tag{34}
\]

\[
c_{ij} = P_{d,ij}(\hat{\lambda}_{ij}^{k+1} - \hat{\lambda}_{ij}^{k}) + A_{ij} \hat{x}_j^{k} + A_{ij} w_i^{k+1}, \quad j \in \mathcal{N}_i, \tag{35}
\]

where (14) is used in deriving (35). Since \( P_{d,ij} \) is a nonsingular matrix, (35) defines a mapping from \( w_i^{k+1} \) to \( \hat{\lambda}_{ij}^{k+1} \):

\[
\hat{\lambda}_{ij}^{k+1} = \hat{\lambda}_{ij}^{k} + P_{d,ij}^{-1}(c_{ij} - A_{ij} \hat{x}_j^{k} - A_{ij} w_i^{k+1}), \quad j \in \mathcal{N}_i. \tag{36}
\]

With this mapping, (34) can then be reformulated as

\[
\sum_{j \in \mathcal{N}_i} A_{ij}^T \left( \hat{\lambda}_{ij}^{k} + P_{d,ij}^{-1}(c_{ij} - A_{ij} \hat{x}_j^{k} - A_{ij} w_i^{k+1}) \right) \in \partial_{w_i} f_i(w_i^{k+1}). \tag{37}
\]

By inspection of (37), it can be shown that (37) is in fact an optimality condition for the following optimization problem

\[
w_i^{k+1} = \arg \min_{w_i} \left[ f_i(w_i) + \frac{1}{2} \| c_{ij} - A_{ij} \hat{x}_j^{k} - A_{ij} w_i \|^2_{P_{d,ij}^{-1}} - w_i^T \sum_{j \in \mathcal{N}_i} A_{ij}^T \hat{\lambda}_{ij}^{k} \right]. \tag{38}
\]

The above analysis suggests that \( \hat{\lambda}_{ij}^{k+1} \) can be alternatively computed through an intermediate quantity \( w_i^{k+1} \). We summarize the result in a proposition below.

**Proposition 1.** Considering a node \( i \in \mathcal{V} \) at iteration \( k \), the new estimate \( \hat{\lambda}_{ij}^{k+1} \) for each \( j \in \mathcal{N}_i \) can be obtained by following (36), where \( w_i^{k+1} \) is computed by (38).

Proposition 1 suggests that the estimate \( \hat{\lambda}_{ij}^{k+1} \) can be easily computed from \( w_i^{k+1} \). We argue in the following that the point-to-point transmission of \( \{\hat{\lambda}_{ij}^{k+1}, j \in \mathcal{N}_i\} \) can be replaced with broadcast transmission of \( w_i^{k+1} \).
TABLE I
Synchronous PDMM WHERE FOR EACH $i \in V$, $P_{d,ij} = P_{p,ij}^{-1}$.

| Initialization: $\{x_i\}$ and $\{\lambda_{ij}\}$ |
| Repeat |
| for all $i \in V$ do |
| $\hat{x}_i^{k+1} = \arg\min_{x_i}[f_i(x_i) - x_i^T(\sum_{j \in N_i} A_{ij}^T \hat{\lambda}_{ij}^{k}) + \sum_{j \in N_i} \frac{1}{2} \|A_{ij}x_i + A_{ji} \hat{x}_j^k - c_{ij}\|_2^2_P_{p,ij}]$ |
| end for |
| for all $i \in V$ and $j \in N_i$ do |
| $\hat{\lambda}_{ij}^{k+1} = \hat{\lambda}_{ji}^{k} + P_{p,ij}(c_{ij} - A_{ij} \hat{x}_j^k - A_{ji} \hat{x}_i^{k+1})$ |
| end for |
| $k \leftarrow k + 1$ |
| Until some stopping criterion is met |

We see from (36) that the computation of the node-specific message $\hat{\lambda}_{ij}^{k+1}$ (from node $i$ to node $j$) only consists of the quantities $w_i^{k+1}$, $x_j^k$, and $\hat{x}_j^k$. Since $\hat{\lambda}_{ji}^{k}$ and $\hat{x}_j^k$ are available at node $j$, the message $\hat{\lambda}_{ij}^{k+1}$ can therefore be computed at node $j$ once the common message $w_i^{k+1}$ is received. In other words, it is sufficient for node $i$ to broadcast both $x_i^{k+1}$ and $w_i^{k+1}$ to all its neighbors. Every node-specific message $\hat{\lambda}_{ij}^{k+1}$, $j \in N_i$, can then be computed at node $j$ alone.

Finally, in order for the broadcast transmission to work, we assume there is no transmission failure between neighboring nodes. The assumption ensures that there is no estimate inconsistency between neighboring nodes, making the broadcast transmission reliable.

2) Reducing two minimizations to one: In the second scenario, we study under what conditions the two minimizations (29)-(30) (which also hold for asynchronous PDMM) reduce to one minimization.

**Proposition 2.** Considering a node $i \in V$ at iteration $k$, if the matrix $P_{d,ij}$ for every neighbor $j \in N_i$ is chosen to be $P_{d,ij} = P_{p,ij}$, then there is $\hat{x}_i^{k+1} = w_i^{k+1}$. As a result, $\hat{\lambda}_{ij}^{k+1} = \hat{\lambda}_{ji}^{k} + P_{p,ij}(c_{ij} - A_{ji} \hat{x}_j^k - A_{ij} \hat{x}_i^{k+1})$ $j \in N_i$. (39)

**Proof.** The proof is trivial. By inspection of (29) and (38) under $P_{d,ij} = P_{p,ij}^{-1}$, $j \in N_i$, we obtain $\hat{x}_i^{k+1} = w_i^{k+1}$. □

Similarly to the first scenario, broadcast transmission is also applicable for the second scenario. Since $\hat{x}_i^{k+1} = w_i^{k+1}$, node $i$ only has to broadcast the estimate $\hat{x}_i^{k+1}$ to all its neighbors. Each message $\hat{\lambda}_{ij}^{k+1}$ from node $i$ to node $j$ can then be computed at node $j$ directly by applying (39). See Table I for the procedure of synchronous PDMM.

V. CONVERGENCE ANALYSIS

In this section, we analyze the convergence rates of PDMM for both the synchronous and asynchronous schemes. Inspired by the convergence analysis of ADMM [27], [28], we construct a special inequality (presented in V-B) for $L_P(x, \lambda)$ and then exploit it to analyze both synchronous PDMM (presented in V-C) and asynchronous PDMM (presented in V-D).

Before constructing the inequality, we first study how to properly choose the matrices in the set $P$ (presented in V-A) in order to enable convergence analysis.

A. Parameter setting

In order to analyze the algorithm convergence later on, we first have to select the matrix set $P$ properly. We impose a condition on each pair of matrices $(P_{p,ij} > 0, P_{d,ij} > 0)$, $(i, j) \in E$, in $L_P$:

**Condition 1.** In the function $L_P$, each matrix $P_{d,ij}$ can be represented in terms of $P_{p,ij}$ as $P_{d,ij} = P_{p,ij}^{-1} + \Delta P_{d,ij}$ $\forall (i, j) \in E$, (40) where $\Delta P_{d,ij} \succeq 0$.

Eq. (40) implies that $P_{p,ij}$ and $P_{d,ij}$ can not be chosen arbitrarily for our convergence analysis. If $P_{p,ij}$ is small, then $P_{d,ij}$ has to be chosen big enough to make (40) hold, and vice versa. One special setup for $(P_{p,ij}, P_{d,ij})$ is to let $P_{d,ij} = P_{p,ij}^{-1}$, or equivalently, $\Delta P_{d,ij} = 0$. This leads to the application of Proposition 2, which reduces two minimizations to one minimization for each activated node.

One simple setup in Condition 1 is to let all the matrices in $P$ take scalar form. That is setting $(P_{p,ij}, P_{d,ij})$, $(i, j) \in E$, to be identity matrices multiplied by positive parameters:

$$(P_{p,ij}, P_{d,ij}) = (\gamma_{p,ij} I_{n_{ij}}, \gamma_{d,ij} I_{n_{ij}})$$ (41)

where $\gamma_{p,ij} > 0$, $\gamma_{d,ij} > 0$ and $\gamma_{d,ij} \gamma_{p,ij} \geq 1$. It is worth noting that matrix form of $(P_{p,ij}, P_{d,ij})$ might lead to faster convergence for some optimization problems.

B. Constructing an inequality

Before introducing the inequality, we first define a new function which involves $\{f_i, i \in V\}$ and their conjugates:

$$p(x, \lambda) = \sum_{i \in V} [f_i(x_i) + f_i^*(A_i^T \lambda_i) - \frac{1}{2} \sum_{j \in N_i} e_{ij}^T \lambda_{ij}^{k+1}].$$ (42)

By studying (7) and (13) at a saddle point $(x^*, \lambda^*)$ of $L_P$, one can show that $p(x^*, \lambda^*) = 0$.

With $p(x, \lambda)$, the inequality for $L_P$ can be described as:

**Lemma 4.** Let $(x^*, \lambda^*)$ be a saddle point of $L_P$. Then for any $(x, \lambda)$, there is

$$0 \leq \sum_{i \in V} \sum_{j \in N_i} [\lambda_{ij} - \lambda_{ij}^*]^T (A_{ij} x_i - \frac{c_{ij}}{2}) - (x_i - x_i^*)^T A_{ij}^T \lambda_{ij}^{k+1} + p(x, \lambda),$$ (43)

where equality holds if and only if $(x, \lambda)$ satisfies

$$0 \in \partial_x f_i(x_i) - \sum_{j \in N_i} A_{ij}^T \lambda_{ij}^{k+1} \quad \forall i \in V$$ (44)

$$0 \in \partial_x f_i(x_i) - \sum_{j \in N_i} A_{ij}^T \lambda_{ij}^* \quad \forall i \in V.$$ (45)
Proof. Given a saddle point \((x^{\ast}, \lambda^{\ast})\) of \(L_{P}\), the right hand side of the inequality (43) can be reorganized as
\[
\sum_{i \in V} \left[ \left( -\lambda_{i j}^{\ast T} A_{i j} x_{j} - \frac{c_{i j}}{2} \right) + \lambda_{i j}^{\ast T} A_{i j}^{T} x_{i} \right] + f_{i}(x_{i}) + f_{i}^{\ast}(A_{i}^{T} \lambda_{i})
\]
\[
= \sum_{i \in V} \left[ \left( -\lambda_{i j}^{\ast T} A_{i j} x_{i} + (A_{i j} x_{j}^{\ast} - c_{i j})^{T} \lambda_{i j}^{\ast} \right) + f_{i}(x_{i}) + f_{i}^{\ast}(A_{i}^{T} \lambda_{i}) + \frac{1}{2} \sum_{j \in N_{i}} c_{i j}^{2} \lambda_{i j}^{\ast} \right]
\]
\[
= \sum_{i \in V} \left[ \left( -\lambda_{i j}^{\ast T} A_{i j} x_{i} - \lambda_{i j}^{\ast T} A_{i j}^{T} \lambda_{i j}^{\ast} \right) + f_{i}(x_{i}) + f_{i}^{\ast}(A_{i}^{T} \lambda_{i}) + \frac{1}{2} \sum_{j \in N_{i}} c_{i j}^{2} \lambda_{i j}^{\ast} \right], \tag{46}
\]
where the last equality is obtained by using \((x^{\ast}, \lambda^{\ast}) \in (X, \Lambda)\). Using Fenchel’s inequalities (12), we conclude that for any \(i \in V\), the following two inequalities hold
\[
f_{i}^{\ast}(A_{i}^{T} \lambda_{i}) - \lambda_{i j}^{\ast T} A_{i j}^{T} \lambda_{i j}^{\ast} \geq -f_{i}(x_{i}^{\ast}) \tag{47}
\]
\[
f_{i}(x_{i}) - \lambda_{i j}^{\ast T} A_{i j} x_{i} \geq -f_{i}^{\ast}(A_{i}^{T} \lambda_{i}) \tag{48}
\]
Finally, combining (46)-(48) and the fact that \(p(x^{\ast}, \lambda^{\ast}) = 0\) produces the inequality (43). The equality holds if and only if (47)-(48) hold, of which the optimality conditions are given by (44)-(45) (see (4)-(6) for the argument).

Lemma 4 shows that the quantity on the right hand side of (43) is always lower-bounded by zero. In the next two subsections, we will construct proper upper bounds for the quantity by replacing \((x, \lambda)\) with real estimate of PDMM. The algorithmic convergence will be established by showing that the upper bounds approach to zero when iteration increases.

The conditions (44)-(45) in Lemma 4 are not sufficient for showing that \((x, \lambda)\) is a saddle point of \(L_{P}\). The primal and dual feasibilities \(x \in X\) and \(\lambda \in \Lambda\) are also required to complete the argument, as shown in Lemma 5, 6 and 7 below. Lemma 5 and 6 are preliminary to show that \((x, \lambda)\) is a saddle point of \(L_{P}\) as presented in Lemma 7. These three lemmas will be used in the next two subsections for convergence analysis.

Lemma 5. Let \((x^{\ast}, \lambda^{\ast})\) be a saddle point of \(L_{P}\). Given \(x = x^{\prime}\) which satisfies (45) and \(x^{\prime} \in X\), then \((x^{\prime}, \lambda^{\ast})\) is a saddle point of \(L_{P}\).

Proof. By using (45) and the fact that \(x^{\prime} \in X\) and \(\lambda^{\ast} \in \Lambda\), it is immediate from (24)-(26) that \((x^{\prime}, \lambda^{\ast})\) is a saddle point of \(L_{P}\).

Lemma 6. Let \((x^{\ast}, \lambda^{\ast})\) be a saddle point of \(L_{P}\). Given \(\lambda = \lambda^{\prime}\) which satisfies (44) and \(\lambda^{\prime} \in \Lambda\), then \((x^{\ast}, \lambda^{\prime})\) is a saddle point of \(L_{P}\).

Proof. The proof is similar to that for Lemma 5.

Lemma 7. Let \((x^{\ast}, \lambda^{\ast})\) be a saddle point of \(L_{P}\). Given \((x, \lambda) = (x^{\prime}, \lambda^{\prime})\) which satisfy (44)-(45) and \((x^{\prime}, \lambda^{\prime}) \in (X, \Lambda)\), then \((x^{\prime}, \lambda^{\ast})\) is a saddle point of \(L_{P}\).

Proof. It is known from Lemma 5 and 6 that in addition to \((x^{\ast}, \lambda^{\ast})\), \((x^{\prime}, \lambda^{\ast})\) and \((x^{\prime}, \lambda^{\ast})\) are also the saddle points of \(L_{P}\). By using a similar argument as the one for Lemma 3, one can show that \((x^{\prime}, \lambda^{\ast})\) is a saddle point of \(L_{P}\).

C. Synchronous PDMM

In this subsection, we show that the synchronous PDMM converges with the sub-linear rate \(O(K^{-1})\). In order to obtain the result, we need the following two lemmas.

Lemma 8. Let \((x^{\ast}, \lambda^{\ast})\) be a saddle point of \(L_{P}\). The estimate \((\hat{x}^{k+1}, \hat{\lambda}^{k+1})\) is obtained by performing (29)-(30) under Condition 1. Then there is
\[
\sum_{i \in V} \left[ \left( \lambda_{i j}^{k+1} - \lambda_{i j}^{k} \right)^{T} (A_{i j} \hat{x}^{k+1} - \lambda_{i j}^{k}) - \left( \hat{x}^{k+1} - x_{i}^{k} \right)^{T} \right] + p(\hat{x}^{k+1}, \hat{\lambda}^{k+1}) \leq \sum_{i \in V} \sum_{j \in N_{i}} d_{i j}^{k+1} \tag{49}
\]
where \(d_{i j}^{k+1}\) is given by
\[
d_{i j}^{k+1} = \frac{1}{2} \left[ \sum_{p \neq i, j} \left( p_{i j}^{2} \left( A_{i j}(\hat{x}^{k+1}_{j} - x_{j}^{k}) + p_{j i}^{2} (\lambda^{k+1}_{i j} - \lambda^{k}_{i j}) \right)^{2} + \sum_{d, \delta} (\Delta x^{k}_{i j} - \lambda_{i j}^{k}) \right)^{2} \right] \tag{50}
\]
Proof. See the proof in Appendix A.

Lemma 9. Every pair of estimates \((\hat{x}^{k+1}_{i j}, \hat{\lambda}^{k+1}_{i j})\), \(i \in V\), \(j \in N_{i}\), \(k \geq 0\), in Lemma 8 is upper bounded by a constant \(M\) under a squared error criterion:
\[
\sum_{i \in V} \sum_{j \in N_{i}} \left( p_{i j}^{2} \left( A_{i j}(\hat{x}^{k+1}_{j} - x_{j}^{k}) + p_{j i}^{2} (\lambda^{k+1}_{i j} - \lambda^{k}_{i j}) \right)^{2} \right) \leq M. \tag{51}
\]
Proof. One can first prove (51) for \(k = 0\) by performing algebra on (49)-(50). The inequality (51) for \(k > 0\) can then be proved recursively.

Upon obtaining the results in Lemma 8 and 9, we are ready to present the convergence rate of synchronous PDMM.

Theorem 2. Let \((\hat{x}^{k}, \hat{\lambda}^{k})\), \(k = 1, \ldots, K\), be obtained by performing (29)-(30) under Condition 1. The average estimate \((\bar{x}^{K}, \bar{\lambda}^{K}) = (\frac{1}{K} \sum_{k=1}^{K} \hat{x}^{k}, \frac{1}{K} \sum_{k=1}^{K} \hat{\lambda}^{k})\) satisfies
\[
0 \leq \sum_{i \in V} \sum_{j \in N_{i}} \left( \lambda_{i j}^{K} - \lambda_{i j}^{k} \right)^{T} (A_{i j} \bar{x}^{K}_{j} - \frac{c_{i j}}{2}) - (\bar{x}^{K}_{i} - x_{i}^{k})^{T} \right] + p(\bar{x}^{K}, \bar{\lambda}^{K}) \leq O\left( \frac{1}{K} \right) \tag{52}
\]
Finally, since the left hand side of (54) is a convex function of \((x, \lambda)\), applying Jensen’s inequality to (54) and using the inequality of Lemma 4 yields (52). Similarly, applying Jensen’s inequality to (54) and using the upper-bound result of Lemma 9 yields the asymptotic result (53).

Finally, we use the results of Theorem 2 to show that as \(K\) goes to infinity, the average estimate \((\bar{x}^K, \bar{\lambda}^K)\) converges to a saddle point of \(L_p\).

**Theorem 3.** The average estimate \((\bar{x}^K, \bar{\lambda}^K)\) of Theorem 2 converges to a saddle point \((x^*, \lambda^*)\) of \(L_p\) as \(K\) increases.

**Proof.** The basic idea of the proof is to investigate if \((\bar{x}^K, \bar{\lambda}^K)\) satisfies all the conditions of Theorem 7. By investigation of Lemma 4 and (52), it is clear that the average estimate \((\bar{x}^K, \bar{\lambda}^K)\) asymptotically satisfies the conditions (44)-(45) by letting \((x, \lambda) = (\bar{x}^K, \bar{\lambda}^K)\).

Next we show that as \(K\) increases, \(\bar{x}^K\) asymptotically converges to an element of the primal feasible set \(X\) and so does \(\bar{\lambda}^K\) to an element of the dual feasible set \(\Lambda\). To do so, we reconsider (53) for each pair of directed edges \([i, j]\) and \([j, i]\), which can be expressed as

\[
\lim_{K \to \infty} \left[ P^2_{p,ij}(A_{ij}\bar{x}^K_i + A_{ji}\bar{x}^K_j - c_{ij}) + P^2_{p,ij}(\hat{\lambda}^K_{ji} - \bar{\lambda}^K_{ji}) \right] = 0 \quad \forall [i, j] \in E.
\]

Combining the above two expressions we obtain

\[
\lim_{K \to \infty} A_{ij}\bar{x}^K_i + A_{ji}\bar{x}^K_j = c_{ij} \quad \forall (i, j) \in E.
\]

It is straightforward from Lemma 7 that \((\bar{x}^K, \bar{\lambda}^K)\) converges to a saddle point of \(L_p\) as \(K\) increases.

Further we have the following result from Theorem 3:

**Corollary 1.** If for certain \(i \in V\), the estimate \(\bar{x}^k_i\) in Theorem 2 converges to a fixed point \(x^*_i\) (\(\lim_{K \to \infty} \bar{x}^K_i = x^*_i\)), we have \(\bar{x}^*_i = x^*_i\) which is the \(i\)th component of the optimal solution \(x^*\) in Theorem 3. Similarly, if the estimate \(\hat{\lambda}^k_{ij}\) converges to a point \(\lambda^*_{ij}\), we have \(\hat{\lambda}^*_{ij} = \lambda^*_{ij}\).

**D. Asynchronous PDMM**

In this subsection, we characterize the convergence rate of asynchronous PDMM. In order to facilitate the analysis, we consider a predefined node-activation strategy (no randomness is involved). We suppose at each iteration \(k\), the node \(i = \text{mod}(k, m) + 1\) is activated for parameter-updating, where \(m = |V|\) and \(\text{mod}(\cdot, \cdot)\) stands for the modulus operation. Then, naturally, after a segment of \(m\) consecutive iterations, all the nodes will be activated sequentially, one node at each iteration.

To be able to derive the convergence rate, we consider segments of iterations, i.e., \(k \in \{lm, lm + 1, \ldots, (l+1)m - 1\}\), where \(l \geq 0\). Each segment \(l\) consists of \(m\) iterations. With the mapping \(i = \text{mod}(k, m) + 1\), it is immediate that \(k = ml\) activates node 1 and \(k = (l + 1)m - 1\) activates node \(m\). Based on the above analysis, we have the following result.

**Lemma 10.** Let \(k_1, k_2\) be two iteration indices within a segment \(\{lm, lm + 1, \ldots, (l+1)m - 1\}\). If \(k_1 < k_2\), then \(i_1 < i_2\), where the node-index \(i_q = \text{mod}(k_q, m) + 1, q = 1, 2\).

Upon introducing Lemma 10, we are ready to perform convergence analysis.

**Lemma 11.** Let \((x^*, \lambda^*)\) be a saddle point of \(L_p\). A segment of estimates \(\{\tilde{x}^{k,l}, \tilde{\lambda}^{k,l}\}_{k \in \{lm, \ldots, (l+1)m - 1\}}\), is obtained by performing (31)-(32) under Condition 1. Then there is

\[
\sum_{i \in V \setminus \{u\}} \left[ \tilde{\lambda}^{(l+1)m}_{ij} - \lambda^*_{ij} \right]^T(A_{ij}\tilde{x}^{(l+1)m}_{j} - c_{ij}) + p \left(\tilde{x}^{(l+1)m}_{i} - x^*_{i}\right)^T \cdot A_{ij}\tilde{\lambda}^{(l+1)m}_{ij} \leq \sum_{(u,v) \in E} d^{l+1}_{uv}, \quad (55)
\]

where \(d^{l+1}_{uv}\) is given by

\[
d^{l+1}_{uv} = \frac{1}{2} \left(\|P^2_{p,uv}A_{uv}(\tilde{x}^{lm}_{u} - x^*_{u}) + P^2_{p,uv}(\lambda^*_{u,v} - \tilde{\lambda}^{lm}_{uv})\|^2 - \|P^2_{p,uv}A_{uv}(\tilde{x}^{(l+1)m}_{u} - x^*_{u}) + P^2_{p,uv}(\lambda^*_{u,v} - \tilde{\lambda}^{(l+1)m}_{uv})\|^2 \right)
\]

**Proof.** See the proof in Appendix B. Lemma 10 will be used in the proof to simplify mathematical derivations.

**Remark 2.** We note that Lemma 11 corresponds to Lemma 8 which is for synchronous PDMM. The right hand side of (55) consists of \(|E|\) quantities \(\{d^{l+1}_{uv}\}\) (one for each edge \((u, v) \in E)\)
as opposed to that of (49) which consists of \(\{a_{ij}^k \}_j\) quantities \(\{a_{ij}^{k+1} \}_j\) (one for each directed edge \([i, j] \in \mathcal{E}\)).

**Lemma 12.** Every pair of estimates \((\hat{x}_v^{(l+1)m}, \hat{x}_u^{(l+1)m})\), \((u, v) \in \mathcal{E}, u < v, l \geq 0\), in Lemma 11 is upper bounded by a constant \(M\) under a squared error criterion:

\[
\|P_{p,uv}^\frac{1}{2} A_{uv}(\hat{x}_v^{(l+1)m} - x_v^*) + P_{p,uv}^\frac{1}{2} \lambda_{uv}^* (\lambda_{uv}^* - (\hat{x}_u^{(l+1)m}))\|^2 \leq M.
\]

**Theorem 4.** Let the \(K \geq 1\) segments of estimates \(\{\hat{x}_v^{k+1}, \hat{x}_u^{k+1}\}_k\) for each directed edge \([i, j] \in \mathcal{E}\) be obtained by performing (31)-(32) under Condition 1. The average estimates \((\hat{x}_v^K, \hat{x}_u^K)\) of \((\hat{x}_v^{k+1}, \hat{x}_u^{k+1})\) satisfies

\[
0 \leq \sum_{i \in N_v} \sum_{j \in N_i} \left[ (\lambda_{ij}^K - \lambda_{iu}^k)^T (A_{ij} x_{x}^K - x_i^*) - (\hat{x}_i^K - x_i^*) \right]^T A_{ij}^T \lambda_{ij}^K (p_{uv} x_{x}^K + A_{uv} x_{x}^K - c_{uv})
\]

\[
= \|P_{p,uv}^\frac{1}{2} A_{uv}(\hat{x}_v^K - x_v^*) + P_{p,uv}^\frac{1}{2} \lambda_{uv}^* (\lambda_{uv}^* - (\hat{x}_u^K - x_u^*))\|^2 \leq \|A_v^T \lambda_v^*\| \leq \frac{1}{K}\]

\[
\lim_{K \to \infty} \left[ P_{p,uv}^\frac{1}{2} A_{uv}(\hat{x}_v^K - x_v^*) + P_{p,uv}^\frac{1}{2} \lambda_{uv}^* (\lambda_{uv}^* - (\hat{x}_u^K - x_u^*))\right] = 0 \forall (u, v) \in \mathcal{E}, u < v.
\]

**Proof.** The proof is similar to that of Theorem 2. \(\square\)

Similarly to synchronous PDMM, by using the results of Theorem 4, we can conclude that:

**Theorem 5.** The average estimate \((\hat{x}_v^K, \hat{x}_u^K)\) of Theorem 4 converges to a saddle point \((x^*, \lambda^*)\) of \(L_P\) as \(K\) increases.

**Corollary 2.** If for certain \(u \in \mathcal{V}\), the estimate \(\hat{x}_u^{lm}\) in Theorem 4 converges to a fixed point \(x_u^*\), \(\lim_{|N| \to \infty} \hat{x}_u^{lm} = x_u^*\), we have \(x_u^* = \lambda_{uv}^*\), which is the \(v\)th component of the optimal solution \(x^*\) in Theorem 5. Similarly, if the estimate \(\lambda_{uv}^{lm}\) converges to a point \(\lambda_{uv}^*\), we have \(\lambda_{uv}^* = \lambda_{uv}^*\).

**VI. APPLICATION TO DISTRIBUTED AVERAGING**

In this section, we consider solving the problem of distributed averaging by using PDMM. Distributed averaging is one of the basic and important operations for advanced distributed signal processing [5], [15].

**A. Problem formulation**

Suppose every node \(i\) in a graph \(G = (\mathcal{V}, \mathcal{E})\) carries a scalar parameter, denoted as \(t_i\), \(t_i\) may represent a measurement of the environment, such as temperature, humidity or darkness. The problem is to compute the average value \(t_{ave} = \frac{1}{n} \sum_{i \in \mathcal{V}} t_i\) iteratively only through message-passing between neighboring nodes in the graph.

The above averaging problem can be formulated as a quadratic optimization over the graph as

\[
\min_{\{x_i\}} \sum_{i \in \mathcal{V}} \frac{1}{2} (x_i - t_i)^2 \quad \text{s.t.} \quad x_i - x_j = 0 \quad \forall (i, j) \in \mathcal{E}.
\]

The optimal solution equals to \(x_i^* = \ldots = x_m^* = t_{ave}\), which is the same as the averaging value.

The quadratic problem (60) is inline with (7) by letting

\[
f_i(x_i) = \frac{1}{2} (x_i - t_i)^2 \quad \forall i \in \mathcal{V}
\]

\[
(A_{ij}, A_{ji}, c_{ij}) = (1, -1, 0) \quad \forall (i, j) \in \mathcal{E}, i < j.
\]

In next subsection, we apply PDMM for distributed averaging.

**B. Parameter computations and transmissions**

Before deriving the updating expressions for PDMM, we first configure the set \(P \in \mathcal{P}\). For distributed averaging, all the matrices in \(P\) become scalars. For simplicity, we set the value of the primal scalars and the dual scalars as

\[
P_{p,ij} = \gamma_p \quad \forall (i, j) \in \mathcal{E}
\]

\[
P_{d,ij} = \gamma_d \quad \forall (i, j) \in \mathcal{E},
\]

where the two parameters \(\gamma_p > 0\) and \(\gamma_d > 0\).

We start with the synchronous PDMM. By inserting (61)- (63) into (29), (36) and (38), the updating expression for \((\hat{x}_v^{k+1}, \hat{x}_u^{k+1})\) at iteration \(k\) can be derived as

\[
\hat{x}_v^{k+1} = \frac{t_i + \sum_{j \in N_v} (\gamma_p \hat{x}_j^k + A_{ij} \hat{x}_{ji}^k)}{1 + |N_v| \gamma_p} \quad \forall i \in \mathcal{V}
\]

\[
\hat{x}_u^{k+1} = \frac{1 - \gamma_d}{\gamma_d} (A_{ij} \hat{x}_j^k + A_{ji} \hat{x}_{ji}^k) \quad \forall (i, j) \in \mathcal{E},
\]

\[
\lambda_{ij}^{k+1} = \lambda_{ij}^k \quad \forall (i, j) \in \mathcal{E},
\]

where

\[
u_{ij}^{k+1} = \sum_{j \in N_i} (\hat{x}_{ij}^k + \gamma_d A_{ij} \hat{x}_{ji}^k) + \gamma_d t_i \quad \forall (i, j) \in \mathcal{E},
\]

For the case that \(\gamma_d = \gamma_p^{-1}\), it is immediate from (64) and (66) that \(x_i^{k+1} = u_{i}^{k+1}\), which coincides with Proposition 2.

The asynchronous PDMM only activates one node per iteration. Suppose node \(i\) is activated at iteration \(k\). Node \(i\) then updates \(\hat{x}_i\) and \(\lambda_{ij}, j \in N_i\), by following (64)-(65) while all other nodes remain silent. After computation, node \(i\) then sends \((\hat{x}_i, \lambda_{ij})\) to its neighboring node \(j\) for all neighbors.

As described in Subsection IV-C, if no transmission fails in the graph, the transmission of \(\lambda_{ij}, j \in N_i\), can be replaced by broadcast transmission of \(u_i\) as given by (66). Once \(u_i\) is received by a neighboring node \(j\), \(\lambda_{ij}\) can be easily computed by node \(j\) alone using \(u_i, \hat{x}_j\) and \(\lambda_{ii}\) (see Eq. (65)). If instead the transmission is not reliable, we have to return to point-to-point transmission.

**C. Experimental results**

We conducted three experiments for PDMM applied to distributed averaging. In the first experiment, we evaluated how different parameter-settings w.r.t. \((\gamma_p, \gamma_d)\) affect the convergence rates of PDMM. In the second experiment, we tested the non-perfect channels for PDMM, which lacks theoretical analysis at the moment. Finally, we evaluated the convergence rates of PDMM, ADMM and two gossip algorithms.

The tested graph in the three experiments was a \(10 \times 10\) two-dimensional grid (corresponding to \(m = 100\)), implying that each node may have two, three or four neighbors. The
suggests that synchronous and asynchronous schemes. This phenomenon was found that the above setting led to divergence for both convergence. We also tested the setting \(\gamma\) that leads to the fastest convergence lies on the curve for both the synchronous and the asynchronous updating schemes. Further, it appears that the two optimal settings for \(\gamma\) are in a neighborhood.

The two updating schemes are in a neighborhood. □ (symbol represents a particular setting for PDMM under different parameter-settings. Each \(\gamma\) of PDMM is below the nodes were activated sequentially by following the gossip broadcast algorithm randomly activates one node per iteration, and \(\gamma\) is derived at the moment. As discussed in Subsection IV-C, we tested the initial estimate \(x^0\), \(\hat{x}\) was set as \(x^0, \hat{x}^0 = (0, 0)\), \(\hat{\lambda}_i, \hat{\lambda}_j\) from node \(i\) to node \(j \in \mathcal{N}_i\). Due to transmission failure, PDMM was initialized differently from the first experiment. Each time the algorithm was tested, the initial estimate \(\hat{x}, \hat{\lambda}\) was set as \(\hat{x}^0, \hat{\lambda}^0 = (0, 0)\), which guarantees that every node in the graph has access to the initial estimates of neighboring nodes without transmission.

Each curve in the two subplots was obtained by averaging over 100 simulations to mitigate the effect of random transmission losses. It is seen that transmission failure only slows down the convergence speed of the algorithm. The above property is highly desirable in real applications because transmission losses might be inevitable in some networks (e.g., see [29] for investigation of packet-loss over wireless sensor networks in different environments).

Finally, it is observed that for each transmission-loss in subplot (a), the error goes up in the first few hundred of iterations before deceasing. This may because of the special initialization (67). We have tested the initialization \(\{\hat{x}_i = t_i\}\) for 0\% transmission loss, where the MSE decreases along with the iterations monotonically.

3) performance comparison: In this experiment, we investigated the convergence speeds of four algorithms under the condition of no transmission failure. Besides PDMM, we also implemented the broadcast-based algorithm in [14] (referred to as broadcast), the randomized gossip algorithm in [5] (referred to as gossip) and ADMM. Unlike PDMM and ADMM that can work either synchronously or asynchronously, both broadcast and gossip algorithms can only work asynchronously. While broadcast algorithm randomly activates one node per iteration, gossip algorithm randomly activates one edge per iteration for

![Fig. 2. Performance of PDMM for different parameter settings. Each value in subplot (a) represents the number of iterations required for the synchronous PDMM. On the other hand, each value in subplot (b) represents the number of segments of iterations for the asynchronous PDMM, where each segment consists of 100 iterations. The convex curve in each subplot corresponds to \(\gamma_p \gamma_d = 1\).](image1)

![Fig. 3. Performance of synchronous/asynchronous PDMM under different transmission losses (%).](image2)
Similarly, we set the parameter in ADMM to be 1.

In this paper, we have proposed PDMM for iterative optimization over a general graph. The augmented primal-dual Lagrangian function is constructed of which a saddle point provides an optimal solution of the original problem, which leads to the design of PDMM. PDMM performs broadcast transmission under perfect channel and point-to-point transmission under non-perfect channel. We have shown that both the synchronous and asynchronous PDMMs possess a convergence rate of $O(1/K)$ for general closed, proper and convex functions defined over the graph. As an example, we have applied PDMM for distributed averaging, through which properties of PDMM such as proper parameter-selection and resilience against transmission failure are further investigated.

We note that PDMM is natural when performing node-oriented optimization over a graph as compared to ADMM which involves computing the edge variable $z$ introduced in (3). A few applications in [21], [22] and [23] suggest that PDMM is practically promising. While convergence properties of ADMM under different conditions (e.g., strong convexity and/or the gradients being Lipschitz continuous) are well understood, the convergence properties of PDMM for those conditions remain to be discovered.

**VII. Conclusion**

In this paper, we have proposed PDMM for iterative optimization over a general graph. The augmented primal-dual Lagrangian function is constructed of which a saddle point provides an optimal solution of the original problem, which leads to the design of PDMM. PDMM performs broadcast transmission under perfect channel and point-to-point transmission under non-perfect channel. We have shown that both the synchronous and asynchronous PDMMs possess a convergence rate of $O(1/K)$ for general closed, proper and convex functions defined over the graph. As an example, we have applied PDMM for distributed averaging, through which properties of PDMM such as proper parameter-selection and resilience against transmission failure are further investigated.

We note that PDMM is natural when performing node-oriented optimization over a graph as compared to ADMM which involves computing the edge variable $z$ introduced in (3). A few applications in [21], [22] and [23] suggest that PDMM is practically promising. While convergence properties of ADMM under different conditions (e.g., strong convexity and/or the gradients being Lipschitz continuous) are well understood, the convergence properties of PDMM for those conditions remain to be discovered.

**APPENDIX A**

**PROOF FOR LEMMA 8**

Before presenting the proof, we first introduce a basic inequality, which is described in a lemma below:

**Lemma 13.** Let $f_1(x)$ and $f_2(x)$ be two arbitrary closed, proper and convex functions. $x^*$ minimizes the sum of the two functions, i.e., $x^* = \arg\min_x (f_1(x) + f_2(x))$. Then, there is

$$f_1(x) - f_1(x^*) \geq (x^* - x)^T r(x^*) \forall x,$$

where $r(x^*) \in \partial_{x} f_2(x^*)$.

The above inequality is wildly exploited for the convergence analysis of ADMM and its variants [27], [28], [10]. We will also use the inequality in our proof.
Applying (68) to the updating equations (29)-(30) for \((\hat{x}^{k+1}, \hat{\lambda}^{k+1})\), we obtain a set of inequalities for all \((\vec{x}, \vec{\lambda}) \in \mathbb{R}^{n_i} \times \mathbb{R}^{n_j}\) as

\[
\sum_{j \in \mathcal{N}_i} \left[ f_j^\ast(A_j^T \vec{\lambda}^j) - f_j^\ast(A_j^T \hat{\lambda}^j) \right] \geq \sum_{i \in \mathcal{V}} \left[ (P_{d,ij}(c_{ij} - A_{ji} \hat{x}_j^i - A_{ji} \hat{x}_j^k) + \hat{\lambda}_k^j) \right] A_{ij}^T \hat{\lambda}_j^k \cdot (\vec{x}_i - \hat{x}_i^{k+1}) \geq f_i^\ast(\vec{x}_i) - f_i^\ast(\hat{x}_i^{k+1}) \quad \forall i \in \mathcal{V} \quad (69)
\]

Adding (69)-(70) over all \(i \in \mathcal{V}\), and substituting \((\vec{x}, \vec{\lambda}) = (\vec{x}^*, \vec{\lambda}^*)\), the saddle point of \(L_F\), yields

\[
\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i} \left[ (\vec{\lambda}_{ij}^{k+1} - \vec{\lambda}_{ij}^*)^T \left( A_{ji} \vec{x}_j^{k+1} - \frac{c_{ij}}{2} \right) - (\vec{x}_i^{k+1} - \vec{x}^*)^T \cdot A_i^T \hat{\lambda}_j^k \right] + p(\vec{x}^{k+1}, \vec{\lambda}^*) - p(\vec{x}^*, \vec{\lambda}^*) \leq \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i} \left[ (P_{p,ij}(c_{ij} - A_{ji} \vec{x}_j^{k+1} - A_{ji} \hat{x}_j^k) + \hat{\lambda}_j^{k+1}) \right] A_{ij}^T \hat{\lambda}_j^k \cdot (\vec{x}_i^{k+1} - \vec{x}_i) \]

\[
= \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i} \left[ (P_{p,ij} A_j (\vec{x}_j^{k+1} - \vec{x}_j^k) + \hat{\lambda}_j^{k+1}) \right] A_{ij}^T \hat{\lambda}_j^k \cdot (\vec{x}_i^{k+1} - \vec{x}_i) \]

\[
- \sum_{(i,j) \in \mathcal{E}} \left( \| c_{ij} - A_{ji} \vec{x}_j^{k+1} - A_{ji} \hat{x}_j^k \|^2_{P_{d,ij}} + \| \vec{x}_i^{k+1} - \vec{x}_j^{k+1} \|^2_{P_{d,ij}} \right),
\]

(71)

where the last equality follows from the two optimality conditions (25)-(26).

To further simplify (71), one can first insert the alternative expression (40) for every \(P_{d,ij}\) into (71). After that, the expression (49) can be obtained by simplifying the new expression using (25)-(26) and the following identity

\[
(y_1 - y_2)^T (y_3 - y_4) = \frac{1}{2} \left( \| y_1 + y_3 \|^2 - \| y_1 + y_4 \|^2 - \| y_2 + y_3 \|^2 + \| y_2 + y_4 \|^2 \right).
\]

**APPENDIX B**

**PROOF OF LEMMA 11**

The basic idea for the proof is similar to that for Lemma 8 as presented in Appendix A. However, since asynchronous PDMM activates one node \(i \in \mathcal{V}\) per iteration, it is difficult to tell which neighbors of \(i\) have been recently activated and which have not yet. The above difficulty requires careful treatment in the convergence analysis. We sketch the proof in the following for reference.

We focus on the parameter-updating for a particular segment of iterations \(k \in \{ml, ml + 1, \ldots, ml + m - 1\}\), where \(l \geq 0\).
where the function \( g(k, i(k), j) \) is defined as

\[
g(k, i(k), j) = \left( P_{d, i(k)} \left( \lambda_{i(k)}^{(l)} - \lambda_{i(k)}^{(l+1)} \right) + A_{i(k)} \left( \hat{x}_{i(k)}^{(l+1)} - \hat{x}_{i(k)}^{(l)} \right) \right) \cdot T \left( \lambda_{i(k)}^{(l)} - \lambda_{i(k)}^{(l+1)} \right)
\]

where \( lm \leq k < (l + 1)m \) and \( j \in N_k \).

Now we are in a position to analyze the right hand side of (75). By using the fact that each node \( i \) has \( |N_i| \) different functions \( g(k, i(k), j) \), we can conclude that each edge \((u, v) \in E\) is associated with two functions \( g(k_1, u(k_1), v) \) and \( g(k_2, v(k_2), u) \), where iteration \( k_1 \) and \( k_2 \) activate \( u \) and \( v \), respectively. From (75), it is clear that each edge \((u, v) \) is also associated with the other two functions \( \|c_{u v} - A_{u v} \hat{x}_{u v}^{(l+1)} - \hat{x}_{u v}^{(l)} \|^2_{P_{u v}} \) and \( \|\lambda_{u v}^{(l+1)} - \lambda_{u v}^{(l)} \|^2_{P_{u v}} \). We show in the following that the combination of the above four functions for every edge \((u, v) \in E\) is independent of \( k_1 \) and \( k_2 \). In order to do so, we assume \( k_1 < k_2 \) (or equivalently, \( u < v \) from Lemma 10). From (72), we know that \( s(k_1, v) = lm \) and \( s(k_2, u) = (l + 1)m \). Based on the above information, the four functions for \((u, v) \in E\) can be simplified as

\[
g(k_1, u(k_1), v) + g(k_2, v(k_2), u) - \|c_{u v} - A_{u v} \hat{x}_{u v}^{(l+1)} - \hat{x}_{u v}^{(l)} \|^2_{P_{u v}}
- \|c_{u v} - A_{u v} \hat{x}_{u v}^{(l+1)} - \hat{x}_{u v}^{(l)} \|^2_{P_{u v}}
= g(k_1, u(k_1), v) - \|\lambda_{u v}^{(l+1)} - \lambda_{u v}^{(l)} \|^2_{P_{u v}}
- \|c_{u v} - A_{u v} \hat{x}_{u v}^{(l+1)} - \hat{x}_{u v}^{(l)} \|^2_{P_{u v}}
= \left[ P_{u v} \lambda_{u v}^{(l+1)} + A_{u v} \hat{x}_{u v}^{(l+1)} - \hat{x}_{u v}^{(l)} \right] \cdot T \left( \lambda_{u v}^{(l+1)} - \lambda_{u v}^{(l)} \right)
+ \|c_{u v} - A_{u v} \hat{x}_{u v}^{(l+1)} - \hat{x}_{u v}^{(l)} \|^2_{P_{u v}}
\]

(76)

where \( d_{u v}^{l+1} \) is given by (56), of which the derivative is similar to that for \( d_{u v}^{l+1} \) in (50). The term \( u(k_1) \) in (76) is simplified as \( u \) since we already assume that at iteration \( k_1 \), node \( u \) is activated. The quantity \( d_{u v}^{l+1} \) is a function of \( m \) and \( l \) instead of \( k_1 \). Finally, combining (75) and (77) produces (55).

REFERENCES

Guoqiang Zhang received the B.Eng. from University of Science and Technology of China (USTC) in 2003, M.Phil. degree from University of Hong Kong in 2006, and Ph.D. degree from Royal Institute of Technology in 2010. From the spring of 2011, he worked as a Postdoctoral Researcher at Delft University of Technology. From the spring of 2015, he worked as a senior researcher at Ericsson AB, Sweden. Since 2017, he has been a senior lecturer in the school of computing and communications, University of Technology Sydney, Australia. His current research interests include algorithm design and performance analysis for distributed processing, signal processing over wireless sensor networks, and application of distributed optimization in training artificial neural networks.

Richard Heusdens received the M.Sc. and Ph.D. degrees from Delft University of Technology, Delft, The Netherlands, in 1992 and 1997, respectively. Since 2002, he has been an Associate Professor in the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology. In the spring of 1992, he joined the digital signal processing group at the Philips Research Laboratories, Eindhoven, The Netherlands. He has worked on various topics in the field of signal processing, such as image/video compression and VLSI architectures for image processing algorithms. In 1997, he joined the Circuits and Systems Group of Delft University of Technology, where he was a Postdoctoral Researcher. In 2000, he moved to the Information and Communication Theory (ICT) Group, where he became an Assistant Professor responsible for the audio/speech signal processing activities within the ICT group. He held visiting positions at KTH (Royal Institute of Technology, Sweden) in 2002 and 2008 and was a guest professor at Aalborg University from 2014-2016. He is involved in research projects that cover subjects such as audio and acoustic signal processing, speech enhancement, and distributed signal processing.