# *Distributed Alternating Direction Method of Multipliers*

# Distributed Alternating Direction Method of Multipliers*

Ermin Wei† and Asuman Ozdaglar†

***Abstract*— We consider a network of agents that are cooperatively solving a global unconstrained optimization problem, where the objective function is the sum of privately known local objective functions of the agents. Recent literature on distributed optimization methods for solving this problem focused on subgradient based methods, which typically converge at the rate $O\left(\frac{1}{\sqrt{k}}\right)$, where $k$ is the number of iterations. In this paper, we introduce a new distributed optimization algorithm based on Alternating Direction Method of Multipliers (ADMM), which is a classical method for sequentially decomposing optimization problems with coupled constraints. We show that this algorithm converges at the rate $O\left(\frac{1}{k}\right)$.**

## I. Introduction

Some of the problems in machine learning and statistical inference, LASSO (least-absolute shrinkage and selection operator) for instance, can be characterized by a network of agents, where each of the agents is associated with a local cost function and the system objective is to minimize the sum of all local costs. The local cost functions are often determined by a large private data set, which makes passing information regarding the local cost function very difficult. These problems motivated research interest in developing distributed method for solving optimization problems [4], [7], [12], [13], [15], [18], [19].

There are two general types of distributed algorithms. The first type is (sub)gradient based, where at each step a (sub)gradient related step is taken, followed by averaging with neighbors. The computation at each step can be very inexpensive and lead to distributed implementations [4], [9], [8], [11], [13], [14]. The best known rate of convergence for subgradient based distributed methods is $O(\frac{1}{\sqrt{k}})$. The second type of distributed algorithm is for solving constrained problems and it relies on dual methods. In these methods, at each step for a fixed dual variable, the primal variables are solved to minimize some Lagrangian related function, then the dual variables are updated accordingly [3], [6], [7], [17]. This type of methods is preferred when each agent can solve the local optimization problem efficiently. One of the well known method of this kind is the Alternating Direction Method of Multipliers (ADMM), which decomposes the original problem into two sub-problems, sequentially solves them and updates the dual variables associated with a coupling constraint at each iteration. The best known rate of convergence for the classic ADMM algorithm is $O(\frac{1}{k})$.

This work was supported by National Science Foundation under Career grant DMI-0545910 and AFOSR MURI FA9550-09-1-0538.

†Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology

The drawback of the standard ADMM method is that it partitions the problem into only two subproblems and thus cannot be implemented in a distributed way for a larger network. In recent works [5], [16] and [10], distributed ADMM algorithms tailored for specific machine learning problems and parameter estimation in wireless sensor networks have been proposed without convergence rate analysis. In this work, we propose a distributed ADMM algorithm for an unconstrained general optimization problem over an $N$-agent network. We first transform the problem into one with constraints and then distribute the problem to $N$ agents. Each agent implements the algorithm using local cost functions and information obtained via communication with neighbors. We also establish that the proposed algorithm has $O(\frac{1}{k})$ rate of convergence.

In the distributed ADMM algorithm, the updates of the agents are done in a sequential order.[1] In this sense, our proposed method is closely related to incremental distributed algorithms studied in the literature [2], [15], where each agent takes turn to update the system wide decision variable and passes the updated variable to the network. The difference here is that each agent maintains and updates its local estimate. The analysis in this paper is related to [3] and [6]. In both of the works, convergence analysis are done to the standard ADMM algorithm. One of our contributions is to extend the analysis to the distributed ADMM algorithm with $N$ agents.

The rest of the paper is organized as follows. Section II defines the problem formulation and equivalent transformation. In Section III, we review the standard ADMM algorithm and develop the distributed ADMM method. In Section IV, we present the convergence analysis of the distributed ADMM algorithm. Section V contains our concluding remarks.

**Basic Notation and Notions:**

A vector is viewed as a column vector, unless clearly stated otherwise. For a matrix $A$, we write $A_{ij}$ to denote the matrix entry in the $i^{th}$ row and $j^{th}$ column, and $[A]_i$ to denote the $i^{th}$ column of the matrix $A$, and $[A]^j$ to denote the $j^{th}$ row of the matrix $A$. For a vector $x$, $x_i$ denotes the $i^{th}$ component of the vector. We use $x'$ and $A'$ to denote the transpose of a vector $x$ and a matrix $A$ respectively. For a real-valued function $f : X \to \mathbb{R}$, where $X$ is a subset of $\mathbb{R}^n$, the gradient

[1] In the working paper [20], we relax this assumption and the update is determined randomly in a distributed way.

vector of $f$ at $x$ in $X$ are denoted by $\nabla f(x)$. We use standard Euclidean norm unless otherwise noted, i.e., for a vector $x$ in $\mathbb{R}^n$, $||x|| = \left(\sum_{i=1}^n x_i^2\right)^{\frac{1}{2}}$.

## II. Formulation

We consider a network, represented by an undirected connected simple graph with $N$ nodes and $M$ edges, $G = \{V, E\}$, where the set $V$ denotes the set of nodes and the set $E$ denotes the set of undirected edges. The nodes can be considered as agents in many applications, we will use the two terms interchangeably. We assume that nodes are ordered from 1 to $N$ and use $e_{ij}$ to denote the edge between nodes $i$ and $j$ with $i < j$.

Each node is associated with a cost function $f_i : \mathbb{R} \to \mathbb{R}$, which is not necessarily differentiable. The nodes in the network are collectively solving the following unconstrained optimization problem

$$\min_{y \in \mathbb{R}} \quad \sum_{i=1}^{N} f_i(y). \tag{1}$$

We assume that each $f_i$ is locally known by agent $i$ only.[2]

We next consider a reformulation of problem (1) which will be used in the development of our distributed algorithm. In particular, we will introduce a separate decision variable $x_i$ for each of the nodes and impose the constraint $x_i = x_j$ for all pairs $(i, j)$ with $e_{ij}$ in set $E$, which guarantees that the node decision variables are equal, i.e., $x_i = \bar{x}$ for all nodes $i$ for some $\bar{x}$. We write the transformed problem compactly by introducing the *edge-node incidence matrix*, which represents the network topology. The edge-node incidence matrix of network $G$, denoted by $A$, has dimension $M \times N$. Each row of matrix $A$ corresponds to an edge in the graph and each column represent an agent. The row corresponding to the edge $e_{ij}$, which we denote by $[A]^{e_{ij}}$, has 1 in the $i^{th}$ column, $-1$ in the $j^{th}$ column and 0 in the other columns, i.e., [3]

$$[A]_k^{e_{ij}} = \begin{cases} 1 & \text{if} \quad k = i, \\ -1 & \text{if} \quad k = j, \\ 0 & \text{otherwise.} \end{cases}$$

For instance, the edge-node incidence matrix for the network in Figure 1 is given by

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 \end{bmatrix}.$$

[2]The analysis in this paper can be extended to the case when the $f_i$ are multi-dimensional functions. To highlight the main ideas, we restrict attention to the case where the $f_i$ are one-dimensional here.

[3]We adopt the convention that the smaller column has entry 1 and the larger one has $-1$. The analysis holds for other consistent way of writing the edge-node incidence matrix, for example, with $-1$ at $i^{th}$ column and 1 at $j^{th}$ column.

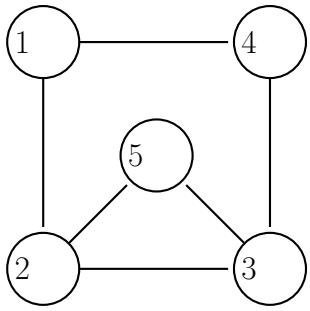


Fig. 1. Sample network topology.

With the edge-node incidence matrix, we can now rewrite problem (1) as

$$\begin{aligned} \min_{x} \quad & \sum_{i=1}^{N} f_i(x_i) \\ s.t. \quad & Ax = 0, \end{aligned} \tag{2}$$

where $x$ is the vector $[x_1, x_2, \ldots, x_N]'$. The constraint $Ax = 0$ is a compact way of writing $x_i = x_j$ for nodes $i$ and $j$ which are connected by an edge. For the rest of the paper, we will focus on solving problem (2). We denote the optimal value of problem (2) by $F^*$.

For notational convenience, we define the function $F : \mathbb{R}^N \to \mathbb{R}$ as

$$F(x) = \sum_{i=1}^{N} f_i(x_i), \tag{3}$$

where $x = [x_1 \ldots, x_N]'$. We denote by $L : \mathbb{R}^N \times \mathbb{R}^M \to \mathbb{R}$ the Lagrangian function given by

$$L(x, \lambda) = F(x) - \lambda' A x, \tag{4}$$

where $\lambda$ in $\mathbb{R}^M$ is the Lagrange multiplier vector associated with the constraint $Ax = 0$.

We impose the following standard assumptions on the optimization problem.

*Assumption 1: (Convexity)* Each of the cost function $f_i : \mathbb{R} \to \mathbb{R}$ is closed, proper and strictly convex.

*Assumption 2: (Existence of a Saddle Point)* The Lagrangian $L(x, \lambda) = F(x) - \lambda' A x$ has a saddle point, i.e., there exists a solution, multiplier pair $(x^*, \lambda^*)$ with

$$L(x^*, \lambda) \leq L(x^*, \lambda^*) \leq L(x, \lambda^*) \tag{5}$$

for all $x$ in $\mathbb{R}^N$ and $\lambda$ in $\mathbb{R}^M$.

Assumption 1 implies that $F(x)$ [cf. Eq. (3)] and therefore $L(x, \lambda^*)$ is strictly convex in $x$. This assumption will be used in our convergence analysis. Assumption 2 implies that problem (2) has an optimal solution, which we denote by $x^*$.

## III. Algorithm

In this section, we develop our distributed optimization algorithm, which is a primal dual algorithm motivated by

the alternating direction method of multipliers (ADMM) (see [3] for a recent survey on ADMM). The standard ADMM algorithm involves decomposing the decision vector into subvectors, updating each subvector sequentially (using the most recent value for the rest of the subvectors) by minimizing an augmented Lagrangian function, and finally updating the Lagrange multiplier corresponding to the constraint that couples the subvectors using a dual subgradient method. The ADMM algorithm has typically been used for solving problems in which the decision variable is decomposed into *two subvectors*, which are coupled with a linear constraint. Hence, it can be viewed as solving a 2-agent special case of problem (2). Here we extend the formulation to $N$ subvectors (each corresponding to an agent) and develop an ADMM that can be implemented in a distributed manner over the connected network in which the agents are situated. We first present the standard ADMM formulation and algorithm and then present our distributed ADMM algorithm.

### A. Preliminaries: Standard ADMM Algorithm

The standard ADMM algorithm solves the following problem[4]

$$\begin{aligned} \min_{y,z} \quad & f(y)+g(z) \\ s.t. \quad & Fy+Dz=c, \end{aligned} \tag{6}$$

for variables $y$ in $\mathbb{R}^n$, $z$ in $\mathbb{R}^m$, matrices $F$ in $\mathbb{R}^{p\times n}$, $D$ in $\mathbb{R}^{p\times m}$ and vector $c$ in $\mathbb{R}^p$. We consider the augmented Lagrangian function given by

$$\begin{aligned} L_\rho(y,z,\mu) = f(y)+g(z)-\mu'(Fy+Dz-c) \\ +\frac{\rho}{2}\left|\left|Fy+Dz-c\right|\right|^2, \end{aligned} \tag{7}$$

where $\mu$ is the Lagrange multiplier corresponding to the constraint $Fy+Dz=c$ and $\rho$ is a positive scalar. The ADMM algorithm updates the primal variables $y$ and $z$ and the Lagrange multiplier $\mu$ as follows: Starting from some initial vector $(y^0,x^0,\mu^0)$[5], at iteration $k\geq 0$, the variables are updated as

$$y^{k+1}=\operatorname*{argmin}_y L_\rho(y,z^k,\mu^k), \tag{8}$$

$$z^{k+1}=\operatorname*{argmin}_z L_\rho(y^{k+1},z,\mu^k), \tag{9}$$

$$\mu^{k+1}=\mu^k-\rho(Fy^{k+1}+Dz^{k+1}-c). \tag{10}$$

Note that stepsize used in updating the Lagrange multiplier vector is the same as the augmented Lagrangian function parameter $\rho$.

The ADMM algorithm has an equivalent representation, which we will use to develop our distributed ADMM algorithm. The equivalent representation is based on the following transformation. The last two terms in the augmented Lagrangian function [cf. Eq. (7)] satisfy $\mu'(Fy+Dz-c)+\frac{\rho}{2}\left|\left|Fy+Dz-c\right|\right|^2=\frac{\rho}{2}\left|\left|\frac{1}{\rho}\mu+Fy+Dz-c\right|\right|^2-\frac{1}{2\rho}\left|\left|\mu\right|\right|^2$, where we used the identity $2a'b+\left|\left|b\right|\right|^2=\left|\left|a+b\right|\right|^2-\left|\left|a\right|\right|^2$ replacing $a=\mu$ and $b=Fy+Dz-c$. Therefore, by ignoring terms which are independent of the minimization variables, updates (8) and (9) can be expressed as

$$y^{k+1}=\operatorname*{argmin}_y f(y)+\frac{\rho}{2}\left|\left|Fy+Dz^k-c+\frac{1}{\rho}\mu^k\right|\right|^2, \tag{11}$$

$$z^{k+1}=\operatorname*{argmin}_z g(z)+\frac{\rho}{2}\left|\left|Fy^{k+1}+Dz-c+\frac{1}{\rho}\mu^k\right|\right|^2. \tag{12}$$

We will now use updates (11), (12) and (10) to develop the distributed ADMM algorithm.

### B. Distributed ADMM Algorithm

In the distributed ADMM algorithm, we associate a dual variable $\lambda_{ij}$ with the constraint $x_i=x_j$ on edge $e_{ij}$, and denote by $\lambda$ the vector of dual variables, i.e., $[\lambda_{ij}]_{ij,e_{ij}\in E}$. Each agent $i$ keeps a local decision estimate $x_i$ in $\mathbb{R}$ and a vector of dual variables $\lambda_{ki}$ with $k<i$. We say agent $i$ *owns* the dual variable $\lambda_{ki}$ for $k<i$ and agent $i$ updates the dual variable.

To facilitate the development of our algorithm, we introduce the following notation. We partition the neighbors of a node $i$ into two sets, which we call *predecessors* and *successors* of node $i$, denoted by $P(i)$ and $S(i)$. The set of the predecessors of $i$ consists of neighbors whose index is smaller than $i$, i.e., $P(i)=\{j \mid e_{ji}\in E, j<i\}$, and successors of $i$ are the neighbors with index larger than $i$, i.e., $S(i)=\{j \mid e_{ij}\in E, i<j\}$. Since the graph $G$ is a simple graph, i.e., no edge of the type $e_{ii}$, the set $P(i)\cup S(i)$ includes all neighbors of node $i$.

Following the development of ADMM described in the previous section, we use an augmented Lagrangian approach to solve problem (2) and use scalar $\beta>0$ as the penalty parameter. We note that each row of the constraint $Ax=0$ corresponds to $x_i-x_j=0$ for some $i<j$ with $e_{ij}$ in the set $E$ and the *distributed ADMM algorithm* for solving problem (2) is given as follows:

A Initialization: choose some arbitrary $x_i^0$ in $\mathbb{R}$ for $i=1,\ldots,N$, which are not necessarily all equal.

B For $k\geq 0$,

a Each agent $i$ updates its estimate $x_i^k$ in a sequential order with

$$\begin{aligned} x_i^{k+1} = \operatorname*{argmin}_{x_i} f_i(x_i) \\ +\frac{\beta}{2}\sum_{j\in P(i)}\left|\left|x_j^{k+1}-x_i-\frac{1}{\beta}\lambda_{ji}^k\right|\right|^2 \\ +\frac{\beta}{2}\sum_{j\in S(i)}\left|\left|x_i-x_j^k-\frac{1}{\beta}\lambda_{ij}^k\right|\right|^2 \end{aligned} \tag{13}$$

[4]This subsection follows closely the development in [3]

[5]We use superscripts to denote the iteration number

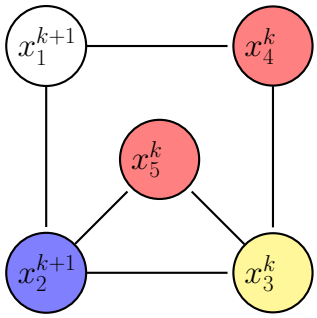


Fig. 2. Sample algorithm evolution.

b Each agent updates $\lambda_{ji}$ that he owns, for all $j$ in $P(i)$,

$$\lambda_{ji}^{k+1} = \lambda_{ji}^k - \beta(x_j^{k+1} - x_i^{k+1}). \tag{14}$$

**Remarks**: We assume that the nodes are ordered. In the above algorithm, at each iteration, the agents update in a sequential way, i.e., at iteration $k$ agent $i$ updates before agent $j$ if $i < j$. In [20] we relax this assumption and the node update is determined randomly in a distributed manner.

We assume that as soon as the variables $x_i^k$ and $\lambda_{ji}^k$ are available to agent $i$, all of its neighbors can also access the data by local information exchange. The above algorithm is well defined, since the step B.a and B.b are implemented in a sequential way, the value of $x_j^{k+1}$ for $j$ in $P(i)$ and $x_j^k$ for $j$ in $S(i)$ are available for agent $i$ during its turn to update. The algorithm can be implemented in a distributed way, since the only communication (of scalar information $x_j^k$ and $\lambda_{ji}^k$) is with immediate neighbors at each step.

This method is related to incremental methods, where each of the agents take turn to update the system wide decision variable and passes the updated variable to the network. The major difference here is that each agent has a local copy of the decision variable, which is not publicly accessible by all the other agents.

For illustration purposes, in Figure 2, we show a snapshot of the distributed ADMM algorithm applied to the network in Figure 1. In the figure, nodes 1 and 2 have completed the $(k+1)^{th}$ updates, and node 3 (highlighted in yellow) is the next one to update. The predecessor of node 3, i.e., $P(3) = \{2\}$, is colored in blue and the successors of node 3, i.e., $S(3) = \{4, 5\}$, are colored in red. Node 3 will use $x_2^{k+1}$, $x_4^k$ and $x_5^k$ to implement update steps B.a and B.b.

## IV. Convergence Analysis

In this section, we present our analysis of the convergence properties of the distributed ADMM algorithm presented in the last section and show that it has a $O(\frac{1}{k})$ rate of convergence, where we count the updates (13) and (14) for all $i$ as one iteration. The analysis here is inspired by the work in [6] and [3], both of which analyzed the standard ADMM algorithm applicable to the special case of a 2-node network. We will first introduce some notation for representational convenience, and derive three lemmas, all of which will then be used to establish the convergence properties of the algorithm.

We denote by $B$, a matrix $B$ in $\mathbb{R}^M \times \mathbb{R}^N$ given by $B = \min\{0, A\}$, where the maximization is taken element-wise. For example, the $B$ matrix for the network given in Figure 1 has the following form,

$$B = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix}.$$

From the definition of matrix $B$, it follows that each row of matrix $B$ corresponds to one edge $(i, j)$ with $i < j$ and consists of exactly one entry of $-1$ at position $j$. We denote by $[B]^{e_{ij}}$ the row of $B$ corresponding to the edge $e_{ij}$.

Our analysis of the convergence relies on a key relation derived from the optimality of $x_i^k$ at each iteration. This is established in the following lemma.

*Lemma 4.1:* Let $\{x^k, \lambda^k\}$ be the iterates generated by our distributed ADMM algorithm for problem (2), with vector $x^k = [x_1^k, x_2^k, \ldots, x_N^k]'$ and vector $\lambda^k = [\lambda_{ij}^k]_{ij,e_{ij}\in E}$. Then the following relation holds for all $k$,

$$\begin{aligned} F(x) - F(x^{k+1}) + (x - x^{k+1})' \big(&-A'\lambda^{k+1} \\ &-\beta A'B(x^{k+1} - x^k) + \beta B'B(x^{k+1} - x^k)\big) \geq 0, \end{aligned} \tag{15}$$

for any $x$ in $\mathbb{R}^N$, where matrix $A$ is the edge-node incidence matrix of the network and matrix $B = \min\{0, A\}$.

*Proof:* We denote by $g_i : \mathbb{R} \to \mathbb{R}$ the function

$$\begin{aligned} g_i^k(x_i) = &\frac{\beta}{2} \sum_{j\in P(i)} \left|\left| x_j^{k+1} - x_i - \frac{1}{\beta}\lambda_{ji}^k \right|\right|^2 \\ &+ \frac{\beta}{2} \sum_{j\in S(i)} \left|\left| x_i - x_j^k - \frac{1}{\beta}\lambda_{ij}^k \right|\right|^2, \end{aligned} \tag{16}$$

and from update (13), we have $x_i^{k+1}$ is the optimizer of $g_i^k + f_i$. The optimality implies there exists some subgradient $h(x_i^{k+1})$ in $\partial f_i(x_i^{k+1})$ such that $h(x_i^{k+1}) + \nabla g_i^k(x_i^{k+1}) = 0$ [1], and thus $(x_i - x_i^{k+1})'[h(x_i^{k+1}) + \nabla g_i^k(x_i^{k+1})] = 0$ for all $x_i$ in $\mathbb{R}$. By definition of subgadient, we have

$$f_i(x_i) \geq f_i(x_i^{k+1}) + (x - x_i^{k+1})' h(x_i^{k+1}).$$

The above two relations yield

$$f_i(x_i) - f_i(x_i^{k+1}) + (x - x_i^{k+1})' \nabla g_i^k(x_i^{k+1}) \geq 0.$$

We then substitute $\nabla g_i^k$ using the definition of function $g$ [cf. Eq. (16)] into the above relation and obtain, $f_i(x_i) - f_i(x_i^{k+1}) + (x_i - x_i^{k+1})' \Big(-\beta \sum_{j\in P(i)} (x_j^{k+1} - x_i^{k+1} - \frac{1}{\beta}\lambda_{ji}^k)$

$+\beta\sum_{j\in S(i)}(x_i^{k+1}-x_j^k-\frac{1}{\beta}\lambda_{ij}^k)\Big)\geq 0$. Using relation (14), we have $f_i(x_i)-f_i(x_i^{k+1})+(x_i-x_i^{k+1})'\Big(\sum_{j\in P(i)}\lambda_{ji}^{k+1}+\sum_{j\in S(i)}-\lambda_{ij}^{k+1}+\sum_{j\in S(i)}\beta(x_j^{k+1}-x_j^k)\Big)\geq 0$. By using the definition of the matrix $A$, we can rewrite the preceding inequality as $f_i(x_i)-f_i(x_i^{k+1})+(x_i-x_i^{k+1})'\Big(-[A]_i'\lambda^{k+1}+\sum_{j\in S(i)}\beta(x_j^{k+1}-x_j^k)\Big)\geq 0$. We then sum the above relation over $i=1,\ldots,N$, and obtain $\sum_{i=1}^N f_i(x_i)-\sum_{i=1}^N f_i(x_i^k)+\sum_{i=1}^N(x_i-x_i^{k+1})'\Big(-[A]_i'\lambda^{k+1}+\sum_{j\in S(i)}\beta(-[B]^{e_{ij}})(x^{k+1}-x^k)\Big)\geq 0$. By using the definition of matrices $A$ and $B$, we can rewrite the terms compactly in matrix representation as $-\sum_{i=1}^N(x_i-x_i^{k+1})'[A]_i'\lambda^{k+1}=-(x-x^{k+1})'A'\lambda^{k+1}$, and $\sum_{i=1}^N(x_i-x_i^{k+1})'\Big(\sum_{j\in S(i)}\beta(x_j^{k+1}-x_j^k)\Big)=\beta(x-x^{k+1})'(-A+B)'B(x^{k+1}-x^k)$. The preceding three relations can be combined to establish the desired relation. ■

We next relate the terms on the left hand side of relation (15) to a combination of vector norms. This is important since we can bound each of the norm term as nonnegative, which will then be used to establish convergence properties by forming a telescoping series.

*Lemma 4.2:* Let $\{x^k,\lambda^k\}$ be the iterates generated by our distributed ADMM algorithm for problem (2), with vector $x^k=[x_1^k,x_2^k,\ldots,x_N^k]'$ and vector $\lambda^k=[\lambda_{ij}^k]_{ij,e_{ij}\in E}$. Let matrix $A$ be the edge-node incidence matrix of the network and matrix $B$ satisfy $B=\min\{0,A\}$. Then the following relation holds for all $k$,

$$\begin{aligned}2(x^{k+1})'A'(\lambda^{k+1}-\lambda^*)+2\beta(x^{k+1})'A'B(x^{k+1}-x^k)\\+2\beta(x^*-x^{k+1})'B'B(x^{k+1}-x^k)=\\\frac{1}{\beta}\left(\left|\left|\lambda^k-\lambda^*\right|\right|^2-\left|\left|\lambda^{k+1}-\lambda^*\right|\right|^2\right)\\+\beta\left(\left|\left|B(x^k-x^*)\right|\right|^2-\left|\left|B(x^{k+1}-x^*)\right|\right|^2\right)\\-\beta\left|\left|B(x^{k+1}-x^k)-Ax^{k+1}\right|\right|^2.\end{aligned}\tag{17}$$

*Proof:* The proof relies on rewriting the terms on the left hand side of (17), using algebraic manipulation and techniques similar to the appendix of [3]. Most of the manipulation are based on two observations: $\lambda^{k+1}=\lambda^k-\beta Ax^{k+1}$ and $||a+b||^2=||a||^2+||b||^2+2a'b$ for arbitrary vectors $a$ and $b$. Due to space constraints, we omit the proof here, interested readers can find the full proof in [20]. ■

We now present a simple lemma based on saddle point properties of the Lagrangian function.

*Lemma 4.3:* Let $(x^*,\lambda^*)$ be a saddle point of the Lagrangian function defined as in Eq. (4). Then

$$Ax^*=0\tag{18}$$

*Proof:* From the definition of a saddle point [cf. Eq. (5)] we have for any variable, multiplier pair $(x,\lambda)$, where $x$ is in $\mathbb{R}^N$ and $\lambda$ is in $\mathbb{R}^M$, the following relation holds

$$F(x^*)-\lambda'Ax^*\leq F(x^*)-(\lambda^*)'Ax^*.$$

The above relation holds for all $\lambda$ and thus $Ax^*=0$. ■

With the three preceding lemmas, we can now establish the main result of the paper, i.e., the $O(\frac{1}{k})$ rate of convergence of the distributed ADMM algorithm.

*Theorem 4.4:* Let $\{x^k,\lambda^k\}$ be the iterates generated by our distributed ADMM algorithm for problem (2), with vector $x^k=[x_1^k,x_2^k,\ldots,x_N^k]'$ and vector $\lambda^k=[\lambda_{ij}^k]_{ij,e_{ij}\in E}$. Let matrix $A$ be the edge-node incidence matrix of the network and matrix $B$ satisfy $B=\min\{0,A\}$. Let $y^k=\frac{1}{k}\sum_{s=0}^{k-1}x^s$ be the ergodic average of $x^k$ up to time $t$, then the following relation holds for all $t$,

$$\begin{aligned}0\leq L(y^k,\lambda^*)-L(x^*,\lambda^*)\leq\\\frac{1}{k}\left(\frac{1}{2\beta}\left|\left|\lambda^0-\lambda^*\right|\right|^2+\frac{\beta}{2}\left|\left|B(x^0-x^*)\right|\right|^2\right).\end{aligned}\tag{19}$$

*Proof:* The first inequality follows immediately from definition of the saddle point of the Lagrangian function [cf. relation (5)].

We now prove the second inequality in Eq. (19). By setting $x=x^*$ in relation (15) from Lemma 4.1, we obtain for all iteration $s$,

$$\begin{aligned}F(x^*)-F(x^{s+1})+(x^*-x^{s+1})'\left(-A'\lambda^{s+1}\right.\\\left.-\beta A'B(x^{s+1}-x^s)+\beta B'B(x^{s+1}-x^s)\right)\geq 0.\end{aligned}$$

Due to feasibility of the optimal solution $x^*$, we have $Ax^*=0$, and the above relation is equivalent to

$$\begin{aligned}F(x^*)-F(x^{s+1})+(x^{s+1})'A'\lambda^{s+1}\\+\left((x^{s+1})'A'\beta B+(x^*-x^{s+1})'\beta B'B\right)(x^{s+1}-x^s)\geq 0.\end{aligned}$$

By adding and subtracting the term $(\lambda^*)'Ax^{s+1}$ from the right hand side of the preceding relation, we have

$$\begin{aligned}F(x^*)-F(x^{s+1})+(x^{s+1})'A'\lambda^*\\(x^{s+1})'A'(\lambda^{s+1}-\lambda^*)+\beta(x^{s+1})'A'B(x^{s+1}-x^s)\\+\beta(x^*-x^{s+1})'B'B(x^{s+1}-x^s)\geq 0,\end{aligned}$$

where we used the identity $a'=a$ for a scalar $a$. We now use Lemma 4.2 to equivalently express the preceding inequality as,

$$\begin{aligned}F(x^*)-F(x^{s+1})+(\lambda^*)'Ax^{s+1}+\frac{1}{2\beta}||\lambda^s-\lambda^*||^2\\+\frac{\beta}{2}||B(x^s-x^*)||^2\geq\frac{1}{2\beta}\left|\left|\lambda^{s+1}-\lambda^*\right|\right|^2\\+\frac{\beta}{2}\left|\left|B(x^{s+1}-x^*)\right|\right|^2\\+\frac{\beta}{2}\left|\left|B(x^{s+1}-x^s)-Ax^{s+1}\right|\right|^2\end{aligned}$$

which holds true for all $s$. We sum the preceding inequality over $s=0,1,\ldots,k-1$ and after telescoping cancelation,

we obtain

$$kF(x^*) - \sum_{s=0}^{k-1} F(x^{s+1}) + (\lambda^*)'A\sum_{s=0}^{k-1} x^{s+1} + \frac{1}{2\beta}\left|\left|\lambda^0 - \lambda^*\right|\right|^2 + \frac{\beta}{2}\left|\left|B(x^0 - x^*)\right|\right|^2 \geq \frac{1}{2\beta}\left|\left|\lambda^k - \lambda^*\right|\right|^2 + \frac{\beta}{2}\left|\left|B(x^k - x^*)\right|\right|^2 + \sum_{s=0}^{k-1}\frac{\beta}{2}\left|\left|B(x^{s+1} - x^s) - Ax^{s+1}\right|\right|^2 \geq 0.$$

Due to the convexity of function $F$, we have $\sum_{s=0}^{k-1} F(x^s) \geq kF(y^k)$, and thus using the definition for $y^k$, we have

$$kF(x^*) - kF(y^k) + (\lambda^*)'Ay^k + \frac{1}{2\beta}\left|\left|\lambda^0 - \lambda^*\right|\right|^2 + \frac{\beta}{2}\left|\left|B(x^0 - x^*)\right|\right|^2 \geq 0.$$

By multiplying both sides by $-k$, we have established the desired relation.

$$F(y^k) - (\lambda^*)'Ay^k - F(x^*) \leq \frac{1}{t}\left(\frac{1}{2\beta}\left|\left|\lambda^0 - \lambda^*\right|\right|^2 + \frac{\beta}{2}\left|\left|B(x^0 - x^*)\right|\right|^2\right).$$

The above relation combined with the definition of the Lagrangian function [cf. Eq. (4)] and Eq. (18) yield the desired relation. ■

Due to strict convexity of the function $F$ [cf. Assumption 1] and linearity of the term $\lambda^* Ax$, we have the function $L(x, \lambda^*)$ is strictly convex in $x$. Therefore we conclude that the function $L(x, \lambda^*)$ has a unique minimizer, i.e., $x^*$ in Assumption 2. Hence Theorem 4.4 guarantees that the value of the Lagrangian function obtained by the ergodic average sequence $y^k$, i.e., $L(y^k, \lambda^*)$, converges to $L(x^*, \lambda^*) = F^*$ at the rate of $O(\frac{1}{k})$. In practice, one might implement the algorithm with one additional variable $y_i^k$ to keep the ergodic average of $x_i^k$ to achieve the $O(\frac{1}{k})$ rate of convergence.[6]

## V. Conclusions

In this paper, we develop a distributed ADMM algorithm for an $N$-agent network, where each agent takes turn to update its local variable. We show that the proposed algorithm achieves a rate of $O(\frac{1}{k})$ rate of convergence, which is the best known rate for dual methods based distributed algorithms. In ongoing work, we are extending this analysis to the algorithm when agents update asynchronously (in some randomized order) and for constrained optimization problems [20].

## References


[1] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[2] D. P. Bertsekas. Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization: A Survey. *Laboratory for Information and Decision Systems Report LIDS-P-2848*, 2010.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*, volume 3(1). Foundations and Trends in Machine Learning, 2010.

[4] J. Duchi, A. Agarwal, and M. Wainwright. Dual averaging for distributed optimization: Convergence and network scaling. *to appear in IEEE Transactions on Automatic Control*, 2012.

[5] P. A. Forero, A. Cano, and G. B. Giannakis. Consensus-based distributed support vector machines. *Journal of Machine Learning Research*, 11:1663–1707, 2010.

[6] B. He and X. Yuan. On the $O(1/t)$ convergence rate of alternating direction method. *Optimization Online*, 2011.

[7] D. Jakovetic, J. Xavier, and J. M. F. Moura. Cooperative convex optimization in networked systems: Augmented Lagrangian algorithms with directed gossip communication. *IEEE Transactions on Signal Processing*, 59(8):3889–3902, 2011.

[8] B. Johansson, T. Keviczky, M. Johansson, and K. H. Johansson. Subgradient methods and consensus algorithms for solving separable distributed control problems. *Proceedings of the* $47^{th}$ *IEEE Conference on Decision and Control*, pages 4185–4190, 2008.

[9] I. Lobel and A. Ozdaglar. Convergence analysis of distributed subgradient methods over random networks. *Proceedings of* $46^{th}$ *Annual Allerton Conference on Communication, Control, and Computing*, pages 353–360, 2008.

[10] G. Mateos and G. B. Giannakis. Distributed recursive least-squares: Stability and performance analysis. *IEEE Transactions on Signal Processing*, 60(7):3740 –3754, 2012.

[11] A. Nedic, Ozdaglar A., and P. Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, 2010.

[12] A. Nedic and A. Ozdaglar. *Convex Optimization in Signal Processing and Communications*, chapter Cooperative distributed multi-agent optimization. Eds., Eldar, Y. and Palomar, D., Cambridge University Press, 2008.

[13] A. Nedic and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[14] S. S. Ram, A. Nedich, and V. V. Veeravalli. Asynchronous gossip algorithms for stochastic optimization. *Proceedings of IEEE Interanational Conference on Decision and Control*, pages 3581–3586, 2009.

[15] S. S. Ram, A. Nedich, and V. V. Veeravalli. Incremental stochastic subgradient algorithms for convex optimization. *SIAM Journal on Optimization*, 20(2):691–717, 2009.

[16] I. D. Schizas, R. Ribeiro, and G. B. Giannakis. Consensus in Ad Hoc WSNs with Noisy Links - Part I: Distributed Estimation of Deterministic Signals. *IEEE Transactions on Singal Processing*, 56:350–364, 2008.

[17] H. Terelius, U. Topcu, and M. Murray. Decentralized multi-agent optimization via dual decomposition. *World Congress of the International Federation of Automatic Control (IFAC)*, 2011.

[18] K. Tsianos and M. Rabbat. Distributed consensus and optimization under communication delays. *to appear in Proceedings* $49^{th}$ *Allerton Conference on Communication, Control and Computing*, 2011.

[19] J. N. Tsitsiklis. *Problems in Decentralized Decision Making and Computation*. PhD thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1984.

[20] E. Wei and A. Ozdaglar. Distributed Alternating Direction Method of Multipliers. *Working Paper*, 2012.


[6] Note that in each iteration, all agents $1, 2, \ldots, N$ updates. Hence large systems will have slower convergence speeds.