Article

Not peer-reviewed version

Distributed Jacobi-Proximal ADMM for Consensus Convex Optimization

Xian-Hong Xiao , Hui Deng , Yang-Dong Xu *

Posted Date: 31 January 2024

doi: 10.20944/preprints202401.2201.v1

Keywords: Consensus convex optimization problem; Distributed Jacobi-proximal ADMM; Multi-agent system; Logistic regression

Article

Distributed Jacobi-Proximal ADMM for Consensus Convex Optimization

Xian-Hong Xiao, Hui Deng and Yang-Dong Xu *

Department of Mathematics, Chongqing University of Posts and Telecommunications, Chongqing, 400065, China; E-mail addresses: xxh9809@126.com (Xian-Hong Xiao), dhS190602017@163.com (Hui Deng)

* Correspondence: xyd04010241@126.com (Yang-Dong Xu).

Abstract: In this paper, a distributed algorithm is proposed to solve a consensus convex optimization problem. It is a Jacobi-proximal alternating direction method of multipliers with a damping parameter γ in the iteration of multiplier. Compared with existing algorithms, it has the following nice properties: (1) The restriction on proximal matrix is relaxed substantively, thus alleviating the weight of the proximal term. Therefore, the algorithm has a faster convergence speed. (2) The convergence analysis of the algorithm is established for any damping parameter $\gamma \in (0, 2]$, which is larger ones in the literature. In addition, some numerical experiments and an application to a logistic regression problem are provided to validate the effectiveness and the characteristics of the proposed algorithm.

Keywords: consensus convex optimization problem; distributed Jacobi-proximal ADMM; multi-agent system; logistic regression

1. Introduction

Consider the following consensus convex optimization problem:

$$\min_{y} \sum_{i=1}^{n} f_i(y) \tag{1.1}$$

where $y \in \mathbb{R}^m$ is the global optimization variable, n is the number of agents in the multi-agent system and f_i $(i = 1, \cdots, n) : \mathbb{R}^m \to \mathbb{R}$ are convex functions. Each f_i is known only by agent i and the agents cooperatively solve the consensus optimization problem. Many problems encountered in machine learning [1] and power network[2] can be posed in the model (1.1).

There are two types distributed algorithms to solve problem (1.1): continuous-time algorithms [3–6] and discrete-time algorithms, among which, discrete-time algorithms can be divided into primal algorithms and dual algorithms. In primal algorithms, each agent takes a (sub)gradient-related step and averages its local solution with those of neighbors [7–9]. One great advantage of these methods is their low computation burden. But slow convergence and low accuracy are two strikes against it. The typical dual algorithms include augmented Lagrangian method [10] and alternating direction method of multipliers (ADMM) [11–16], in which each agent needs to solve a subproblem at each iteration, which is responsible for high computation burden. However, the characteristic that they can quickly converge to exact optimal solutions can make up for it.

The ADMM algorithm has attracted significant research interests in recent years. With regard to distributed ADMM algorithms, almost all developments begin with transforming problem (1.1) into a equivalent form by introducing local copy x_i for each agent $i = 1, 2, ..., n$, and enforcing consensus $x_1 = x_2 = ... = x_n$. For start networks, the reformulation of problem (1.1) can be shown as follows:

$$\begin{aligned} &\min_{x} f(x) := \sum_{i=1}^{n} f_i(x_i) \\ &\text{subject to } \; x_i = \bar{x}, \;\; \forall i, \end{aligned}$$

 doi:10.20944/preprints202401.2201.v1

where $x = [x_1^T, ..., x_n^T]^T$ and $\bar{x}$ is so-called consensus variable. Considerable attentions have been paid to such formulation, which can be referred to [11,12] for details..

A central agent is required in the start network, and thus algorithms in [11,12] have high communication burden and low fault tolerance. This leads to growing research interests in general connected networks. For general connected networks, the consensus optimization problem (1.1) can be rewritten in the following compact form:

$$\min_x f(x) := \sum_{i=1}^{n} f_i(x_i)$$
$$\text{subject to } Ax = 0 \text{ or } Ax + Bz = 0,$$

where $x = [x_1^T, ..., x_n^T]^T$, A, B are matrices related to network structure and z is the slack variable. For this kind of problems, Wei and Ozdaglar [13] proposed a distributed Gauss-Seidel ADMM algorithm and proved that its convergence rate was $O(1/k)$, where the objective function f_i $(1 = 1, \cdots, n)$ are convex. Based on this algorithm, agents can only update in order. To save the waiting time of agents in [13], Yan[14] raised a parallel ADMM algorithm, which adopts Jacobi iterate method. Besides, some distributed ADMM algorithms for nonconvex but differentiable probelms are also established in[15,16].

In addition to the algorithms in [11–16], several ADMM algorithms can also solve problem (1). These algorithms were originally designed to solve multi-block separable problems, which can be can be cast as

$$\min_x \sum_{i=1}^{n} f_i(x_i)$$
$$\text{subject to } A_1x_1 + ... + A_nx_n = c.$$

where $x = [x_1^T, ..., x_n^T]^T$. A wide variety of the proximal ADMM algorithms were proposed for this kind of formulation. The researches on these algorithms mainly focus on proximal matrix P_i and damping parameter γ. Deng et al.[17] presented a parallel ADMM algorithm and the proximal matrix P_i is required to satisfy $P_i \succ (\frac{n}{2-\gamma} - 1)A_i^T A_i$, where $0 < \gamma < 2$. There are two specific choices for the proximal matrix P_i in[18]: (1) Standard proximal matrix $P_i = \tau_i I$; (2) Linearized proximal matrix $P_i = \tau_i I - A_i^T A_i$. Therefore, the condition in [17] can be reduced to

$$P_i = \begin{cases} \tau_i I, & \tau_i > (\frac{n}{2-\gamma} - 1)\|A_i\|^2, \\ \tau_i I - A_i^T A_i, & \tau_i > \frac{n}{2-\gamma}\|A_i\|^2. \end{cases}$$

Afterwards, Sun and Sun[19] came up with an improved proximal ADMM algorithm with partially parallel splitting, where $P_i = \tau_i I - A_i^T A_i$ and a lower bound of the proximal parameter is given by $\tau_i > \frac{4+\max\{1-\gamma, \gamma^2-\gamma\}}{5}(n-1)\|A_i\|^2$, where $0 < \gamma < \frac{1+\sqrt{5}}{2}$.

Inspired by the works in [13,14,17,19], this paper puts forward a distributed Jacobi-proximal ADMM algorithm to solve the consensus convex optimization problem (1.1) over a general connected network. Compared with the state-of-art ones, the proposed algorithm has the following outstanding features.

(1) Compared with the algorithm in [13], the optimization variables of all agents can be updated simultaneously. Hence, the waiting time is saved.

(2) Compared with [14], only half of dual variables are occupied in the proposed algorithm. Therefore, the communication burden among agents and storage cost for each agent are reduced.

(3) The proximal matrix P_i of the presented algorithm is smaller than those in [17,19]. Thus, the distributed Jacobi-proximal ADMM algorithm is favorable based on the general principle given by Fazel et. al [20], that the proximal matrix P_i should be as small as possible. Besides, the value range of damping parameter γ in the proposed algorithm is larger than that of [19].

 doi:10.20944/preprints202401.2201.v1

The rest of this paper is organized as follows. In Section 2, the equivalent form of the consensus convex optimization problem (1.1) is introduced. In addition, based on this equivalent form, a distributed Jacobi-proximal ADMM algorithm is proposed. Section 3 supplies the convergence analysis of the algorithm. In Section 4, extensive numerical experiments are provided to verify the effectiveness of the proposed algorithm. Moreover, the impacts of the penalty parameter, damping parameter and connectivity ratio on the algorithm are investigated. In Section 5, the proposed algorithm is applied to a logistic regression problem and its numerical results are compared with those in [17]. Finally, the conclusions of this paper are presented in Section 6.

2. Problem Formulation and Distributed Jacobi-Proximal ADMM Algorithm

In this section, some notations related to the network are introduced, and the consensus convex optimization problem (1.1) is represented so that it can be solved by ADMM.

The network topology of the multi-agent system is assumed to be a general undirected connected graph, which is described as $G = \{V, E\}$, where V denotes the set of agents, E denotes the set of the edges and $|V| = n$, $|E| = l$. These agents are arranged from 1 to n. The edge between agents i and j with $i < j$ is represented by (i, j) or e_{ij} and $(i, j) \in E$ means that agents i and j can exchange data with each other. The neighbors of agent i are denoted by $N(i) := \{j \in V \mid (i, j) \in E \text{ or } (j, i) \in E\}$ and $d_i = |N(i)|$.

The *edge-node incidence matrix* of the network G is denoted by $\tilde{A} \in \mathbb{R}^{l\times n}$. The row in $\tilde{A}$ that corresponds to the edge e_{ij} is denote by $[\tilde{A}]^{e_{ij}}$, which is defined by

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

Here, the edges of the network are sorted by the order of their corresponding agents. For instance, the edge-node incidence matrix of the network G in Fig. 1 is given by

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

Figure 1. An example of the network G.

According to the edge-node incidence matrix, the *extended edge-node incidence matrix* A of the network G is given by

$$A := \tilde{A} \otimes I_m = \begin{pmatrix} \tilde{a}_{11} I_m & \cdots & \tilde{a}_{1n} I_m \\ \vdots & \ddots & \vdots \\ \tilde{a}_{l1} I_m & \cdots & \tilde{a}_{ln} I_m \end{pmatrix} \in \mathbb{R}^{ml\times mn},$$

where $\otimes$ denotes the *Kronecker product*. Obviously, A is a block matrix with $l * n$ blocks of $m \times m$ matrix.

 doi:10.20944/preprints202401.2201.v1

By introducing separating decision variable x_i for each agent $i = 1, 2, ..., n$, the consensus convex optimization problem (1.1) has the following form:

$$\begin{aligned} &\min_x f(x) := \sum_{i=1}^{n} f_i(x_i) \\ &\text{subject to } x_i = x_j, \ \forall (i,j) \in E, \end{aligned} \tag{2.1}$$

where $x = [x_1^T, x_2^T, ..., x_n^T]^T \in \mathbb{R}^{nm \times 1}$. Clearly, the problem (2.1) is equivalent to problem (1.1) if G is connected.

With the help of the extended edge-node incidence matrix A, the problem (2.1) can be rewritten in the following compact form:

$$\begin{aligned} &\min_x f(x) \\ &\text{subject to } Ax = 0. \end{aligned} \tag{2.2}$$

Dividing the neighbors $N(i)$ of the agent i into two sets: predecessors $P(i) := \{j \in V \mid (j,i) \in E\}$ and successors $S(i) := \{j \in V \mid (i,j) \in E\}$. The distributed Jacobi-proximal ADMM (DJP-ADMM) algorithm is described as Algorithm 1.

Algorithm 1: Distributed Jacobi-proximal ADMM Algorithm (DJP-ADMM)

Initialization: Choose appropriate P_i, ρ, γ, $\{x_i^0\}$, $\{\lambda_{e_{ji}}^0\}$ and $\{\lambda_{e_{ij}}^0\}$, $i = 1, 2, ..., n$.

$0 \leftarrow k$;

while *some stop criteria are not met* **do**

  for $i \leftarrow 1$ **to** n **do**

    Update x_i^{k+1} by

$$x_i^{k+1} := \arg\min_{x_i} f_i(x_i) + \frac{\rho}{2} \sum_{j \in P(i)} \|x_j^k - x_i - \frac{1}{\rho}\lambda_{e_{ji}}^k\|^2 + \frac{\rho}{2} \sum_{j \in S(i)} \|x_i - x_j^k - \frac{1}{\rho}\lambda_{e_{ij}}^k\|^2 + \frac{1}{2}\|x_i - x_i^k\|_{P_i}^2;$$

    for $j \in P(i)$ **do**

      Update $\lambda_{e_{ji}}^{k+1}$ by

$$\lambda_{e_{ji}}^{k+1} := \lambda_{e_{ji}}^k - \gamma\rho(x_j^{k+1} - x_i^{k+1});$$

  $k \leftarrow k + 1$;

Remark 1. The parallel ADMM algorithm presented in [14] is shown as follows:

$$\begin{aligned} &x_i^{k+1} := \arg\min_{x_i} f_i(x_i) + \frac{\rho}{2} \sum_{j \in N(i)} \|x_j^k - x_i - \frac{1}{\rho}\lambda_{e_{ji}}^k\|^2 + \frac{\rho}{2} \sum_{j \in N(i)} \|x_i - x_j^k - \frac{1}{\rho}\lambda_{e_{ij}}^k\|^2, \\ &\lambda_{e_{ji}}^{k+1} := \lambda_{e_{ji}}^k - \rho(x_j^k - x_i^{k+1}), \ j \in N(i). \end{aligned} \tag{2.3}$$

It is clear that the number of dual variables in (2.3) is twice that in DJP-ADMM. Thus, the communication burden among agents and the storage cost for each agent in Algorithm 1 are smaller than ones in [14].

3. Convergence Analysis

In this section, some important notations and technical lemmas are given. Then, the convergence analysis of Algorithm 1 is investigated.

Let

$$\tilde{L}_- = \tilde{A}^T \tilde{A} \in \mathbb{R}^{n \times n}. \tag{3.1}$$

 doi:10.20944/preprints202401.2201.v1

Remark 2. Hong et al. [16] have pointed out that $\tilde{L}_-$ is the *sign Laplace matrix* of the graph G.

The *extended degree matrix* and *extended sign Laplace matrix* of the network G are denoted by

$$D := \tilde{D} \otimes I_m \in \mathbb{R}^{mn\times mn}, \tag{3.2}$$

$$L_- := \tilde{L}_- \otimes I_m \in \mathbb{R}^{mn\times mn}, \tag{3.3}$$

where $\tilde{D}$ is the *degree matrix* of the graph G.

To simplify the notation, let

$$H = \begin{pmatrix} \frac{1}{2}(P_1 + P_1^T) & & \\ & \ddots & \\ & & \frac{1}{2}(P_n + P_n^T) \end{pmatrix} \in \mathbb{R}^{mn\times mn}, \tag{3.4}$$

and

$$Q = H + \rho\bar{A} \otimes I_m, \tag{3.5}$$

where $\bar{A}$ is the *adjacency matrix* of the graph G. To ensure the convergence of Algorithm 1, it is necessary to make an assumption about the matrix Q, which is shown below.

Assumption 1. *The matrix Q is a positive definite matrix.*

Remark 3. If proximal matrices P_i $(i = 1, \cdots, n)$ are symmetric, then Assumption 1 can be reduced to $P + \rho\bar{A} \otimes I_m$ is a positive definite matrix. Therefore, $P = \rho D = \rho\tilde{D} \otimes I_m$ is a feasible choice, where $\tilde{D}$ is the degree matrix of the graph G. In this case, $P_i = \rho d_i I_m$.

Remark 4. By the definition of Q, the matrix Q is symmetric positive definite under Assumption 1, and thus, there exists a matrix M such that

$$Q = M^T M. \tag{3.6}$$

According to the convexity of the objective function, we have following result.

Lemma 1. *Assume that $\{(x^k, \lambda^k)\}$ is the sequence produced by Algorithm 1 for the problem (2.2), where $x^k = [(x_1^k)^T, (x_2^k)^T, ..., (x_n^k)^T]^T$ and $\lambda^k = [\lambda_{e_{ij}}^k]$, $e_{ij} \in E$. Then one has*

$$\begin{aligned} &f(x) - f(x^{k+1}) - (x - x^{k+1})^T A^T \lambda^{k+1} \\ &+ (x - x^{k+1})^T Q(x^{k+1} - x^k) - \rho(\gamma - 1)(x - x^{k+1})^T L_- x^{k+1} \geq 0, \ \forall x \in \mathbb{R}^{mn}. \end{aligned} \tag{3.7}$$

Proof. Define $g_i \ (i = 1, \cdots, n) : \mathbb{R}^m \to \mathbb{R}$ by

$$g_i^k(x_i) := \frac{\rho}{2} \sum_{j\in P(i)} \|x_j^k - x_i - \frac{1}{\rho}\lambda_{e_{ji}}^k\|^2 + \frac{\rho}{2} \sum_{j\in S(i)} \|x_i - x_j^k - \frac{1}{\rho}\lambda_{e_{ij}}^k\|^2 + \frac{1}{2}\|x_i - x_i^k\|_{P_i}^2.$$

Using the iteration of x in Algorithm 1, one can conclude that x_i^{k+1} is the optimizer of $f_i + g_i^k$, i.e.,

$$x_i^{k+1} := \arg\min_{x_i} f_i(x_i) + g_i^k(x_i).$$

Therefore, there exists a subgradient $h(x_i^{k+1}) \in \partial f_i(x_i^{k+1})$ such that $h(x_i^{k+1}) + \nabla g_i^k(x_i^{k+1}) = 0$. Then

$$(x_i - x_i^{k+1})^T \Big(h(x_i^{k+1}) + \nabla g_i^k(x_i^{k+1})\Big) = 0, \ \ \forall x_i \in \mathbb{R}^m. \tag{3.8}$$

 doi:10.20944/preprints202401.2201.v1

Due to the convexity of f_i, we have

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

This together with (3.8) implies that

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

Substituting the gradient ∇g_i^k of the function g_i^k into the above inequality, we have

$$\begin{aligned} &f_i(x_i) - f_i(x_i^{k+1}) + (x_i - x_i^{k+1})^T \\ &\Big(-\rho \sum_{j\in P(i)} (x_j^k - x_i^{k+1} - \frac{1}{\rho}\lambda_{e_{ji}}^k) + \rho \sum_{j\in S(i)} (x_i^{k+1} - x_j^k - \frac{1}{\rho}\lambda_{e_{ij}}^k) + \frac{1}{2}(P_i + P_i^T)(x_i^{k+1} - x_i^k)\Big) \geq 0. \end{aligned}$$

From the iteration of the multipliers, one can obtain that

$$\begin{aligned} &-\rho \sum_{j\in P(i)} (x_j^k - x_i^{k+1} - \frac{1}{\rho}\lambda_{e_{ji}}^k) \\ &= \sum_{j\in P(i)} \Big(\lambda_{e_{ji}}^k - \rho(x_j^k - x_i^{k+1})\Big) = \sum_{j\in P(i)} \Big(\lambda_{e_{ji}}^k - \gamma\rho(x_j^{k+1} - x_i^{k+1}) + \gamma\rho(x_j^{k+1} - x_i^{k+1}) - \rho(x_j^k - x_i^{k+1})\Big) \\ &= \sum_{j\in P(i)} \Big(\lambda_{e_{ji}}^{k+1} + \gamma\rho(x_j^{k+1} - x_i^{k+1}) - \rho(x_j^{k+1} - x_i^{k+1}) + \rho(x_j^{k+1} - x_i^{k+1}) - \rho(x_j^k - x_i^{k+1})\Big) \\ &= \sum_{j\in P(i)} \Big(\lambda_{e_{ji}}^{k+1} + \rho(\gamma - 1)(x_j^{k+1} - x_i^{k+1}) + \rho(x_j^{k+1} - x_j^k)\Big). \end{aligned}$$

Similarly,

$$\rho \sum_{j\in S(i)} (x_i^{k+1} - x_j^k - \frac{1}{\rho}\lambda_{e_{ij}}^k) = \sum_{j\in S(i)} \Big(-\lambda_{e_{ij}}^{k+1} + \rho(\gamma - 1)(x_j^{k+1} - x_i^{k+1}) + \rho(x_j^{k+1} - x_j^k)\Big).$$

Hence,

$$\begin{aligned} &f_i(x_i) - f_i(x_i^{k+1}) + (x_i - x_i^{k+1})^T (\sum_{j\in P(i)} \lambda_{e_{ji}}^{k+1} - \sum_{j\in S(i)} \lambda_{e_{ij}}^{k+1}) \\ &+ (x_i - x_i^{k+1})^T \Big(\rho(\gamma - 1) \sum_{j\in N(i)} (x_j^{k+1} - x_i^{k+1}) + \rho \sum_{j\in N(i)} (x_j^{k+1} - x_j^k) + \frac{1}{2}(P_i + P_i^T)(x_i^{k+1} - x_i^k)\Big) \geq 0, \end{aligned}$$

By the definition of the matrix A, we simplify the above inequality as follows:

$$\begin{aligned} &f_i(x_i) - f_i(x_i^{k+1}) - (x_i - x_i^{k+1})^T [A]_i^T \lambda^{k+1} \\ &+ (x_i - x_i^{k+1})^T \Big(\rho(\gamma - 1) \sum_{j\in N(i)} (x_j^{k+1} - x_i^{k+1}) + \rho \sum_{j\in N(i)} (x_j^{k+1} - x_j^k) + \frac{1}{2}(P_i + P_i^T)(x_i^{k+1} - x_i^k)\Big) \geq 0. \end{aligned}$$

And then,

$$\begin{aligned} &\sum_{i=1}^n f_i(x_i) - \sum_{i=1}^n f_i(x_i^{k+1}) - \sum_{i=1}^n (x_i - x_i^{k+1})^T [A]_i^T \lambda^{k+1} \\ &+ \sum_{i=1}^n (x_i - x_i^{k+1})^T \Big(\rho(\gamma - 1) \sum_{j\in N(i)} (x_j^{k+1} - x_i^{k+1}) + \rho \sum_{j\in N(i)} (x_j^{k+1} - x_j^k) + \frac{1}{2}(P_i + P_i^T)(x_i^{k+1} - x_i^k)\Big) \geq 0. \end{aligned} \tag{3.9}$$

 doi:10.20944/preprints202401.2201.v1

By the definition of matrices A and D, we have

$$-\sum_{i=1}^{n}(x_i - x_i^{k+1})^T[A]_i^T\lambda^{k+1} = -(x - x^{k+1})^T A^T\lambda^{k+1}, \tag{3.10}$$

$$d_i\sum_{i=1}^{n}(x_i - x_i^{k+1})^T x_i^{k+1} = (x - x^{k+1})^T Dx^{k+1}.$$

In addition,

$$\begin{aligned}
&\sum_{i=1}^{n}\Big((x_i - x_i^{k+1})^T\sum_{j\in N(i)}x_j^{k+1}\Big)\\
&= [(x_1 - x_1^{k+1})^T, \dots, (x_n - x_n^{k+1})^T][\sum_{j\in N(1)}(x_j^{k+1})^T, \dots, \sum_{j\in N(n)}(x_j^{k+1})^T]^T\\
&= [(x_1 - x_1^{k+1})^T, \dots, (x_n - x_n^{k+1})^T](\bar{A}\otimes I_m)x^{k+1}\\
&= (x - x^{k+1})^T(\bar{A}\otimes I_m)x^{k+1},
\end{aligned}$$

where $\bar{A}$ is the adjacency matrix of the graph G. The above two relations indicate that

$$\sum_{i=1}^{n}\Big((x_i - x_i^{k+1})^T(\sum_{j\in N(i)}x_j^{k+1} - d_i x_i^{k+1})\Big) = -(x - x^{k+1})^T(D - \bar{A}\otimes I_m)x^{k+1}$$

Therefore, by the definition of the extended sign Laplace matrix L_-, one can conclude that

$$\sum_{i=1}^{n}\Big((x_i - x_i^{k+1})^T\sum_{j\in N(i)}(x_j^{k+1} - x_i^{k+1})\Big) = \sum_{i=1}^{n}\Big((x_i - x_i^{k+1})^T(\sum_{j\in N(i)}x_j^{k+1} - d_i x_i^{k+1})\Big) = -(x - x^{k+1})^T L_- x^{k+1}. \tag{3.11}$$

Analogously,

$$\sum_{i=1}^{n}\Big((x_i - x_i^{k+1})^T\sum_{j\in N(i)}(x_j^{k+1} - x_j^{k})\Big) = (x - x^{k+1})^T(\bar{A}\otimes I_m)(x^{k+1} - x^{k}).$$

Besides, by the definition of matrix Q, we have

$$\sum_{i=1}^{n}\Big[(x_i - x_i^{k+1})^T\Big(\frac{1}{2}(P_i + P_i^T)(x_i^{k+1} - x_i^{k}) + \rho\sum_{j\in N(i)}(x_j^{k+1} - x_j^{k})\Big)\Big] = (x - x^{k+1})^T Q(x^{k+1} - x^{k}). \tag{3.12}$$

Thus, recalling (3.9)-(3.12), inequality (3.7) holds. □

The non-negative property of the norm is very important in the subsequent analysis of convergence. To this end, certain items in Lemma 1 will be concerted into norm form. To simplify some expressions in the proof of the following lemmas, V^k is denoted by

$$V^k = \frac{1}{2\rho\gamma}\|\lambda^k\|^2 + \frac{1}{2}\|M(x^k - x^*)\|^2, \tag{3.13}$$

where M is defined in (3.6).

Under Assumption 1, we can get the following lemma.

 doi:10.20944/preprints202401.2201.v1

Lemma 2. *Assume that* $\{(x^k, \lambda^k)\}$ *is the sequence produced by Algorithm 1 for the problem (2.2), where* $x^k = [(x_1^k)^T, (x_2^k)^T, ..., (x_n^k)^T]^T$ *and* $\lambda^k = [\lambda_{e_{ij}}^k]$, $e_{ij} \in E$. *Then under Assumption 1, one has the following equality*

$$(x^{k+1})^T A^T \lambda^{k+1} + (x^* - x^{k+1})^T Q(x^{k+1} - x^k) = V^k - V^{k+1} - \frac{\rho\gamma}{2}\|Ax^{k+1}\|^2 - \frac{1}{2}\|M(x^{k+1} - x^k)\|^2, \tag{3.14}$$

where $Q = M^T M$.

Proof. To prove (3.14), we firstly claim that

$$(x^{k+1})^T A^T \lambda^{k+1} = \frac{1}{2\rho\gamma}(\|\lambda^k\|^2 - \|\lambda^{k+1}\|^2) - \frac{\rho\gamma}{2}\|Ax^{k+1}\|^2, \tag{3.15}$$

$$(x^* - x^{k+1})^T Q(x^{k+1} - x^k) = \frac{1}{2}(\|M(x^k - x^*)\|^2 - \|M(x^{k+1} - x^*)\|^2) - \frac{1}{2}\|M(x^{k+1} - x^k)\|^2. \tag{3.16}$$

Indeed, by the iteration of the multiplier: $\lambda^{k+1} = \lambda^k - \gamma\rho Ax^{k+1}$, we know

$$(x^{k+1})^T A^T \lambda^{k+1} = (x^{k+1})^T A^T \lambda^k - \rho\gamma\|Ax^{k+1}\|^2, \tag{3.17}$$

and

$$\frac{1}{2\rho\gamma}(\|\lambda^k\|^2 - \|\lambda^{k+1}\|^2) = (x^{k+1})^T A^T \lambda^k - \frac{\rho\gamma}{2}\|Ax^{k+1}\|^2. \tag{3.18}$$

Therefore, equality (3.17) and (3.18) indicate that equality (3.15) is valid. In addition, by distorting some of the terms, we obtain

$$\|M(x^k - x^*)\|^2 - \|M(x^{k+1} - x^*)\|^2 = \|Mx^k\|^2 - \|Mx^{k+1}\|^2 + 2(Mx^*)^T M(x^{k+1} - x^k),$$

and

$$2(x^* - x^{k+1})^T (M^T M)(x^{k+1} - x^k) = 2(Mx^*)^T M(x^{k+1} - x^k) - 2\|Mx^{k+1}\|^2 + 2(Mx^k)^T Mx^{k+1}.$$

Combining the above two equalities, we yield

$$\begin{aligned}
&2(x^* - x^{k+1})^T (M^T M)(x^{k+1} - x^k) \\
&= \|M(x^k - x^*)\|^2 - \|M(x^{k+1} - x^*)\|^2 + 2(Mx^k)^T Mx^{k+1} - (\|Mx^k\|^2 + \|Mx^{k+1}\|^2) \\
&= \|M(x^k - x^*)\|^2 - \|M(x^{k+1} - x^*)\|^2 - \|M(x^{k+1} - x^k)\|^2.
\end{aligned}$$

Taking into account $Q = M^T M$, we can get the equality (3.16). Consequently, by (3.15) and (3.16), the equality (3.14) holds. □

With the help of the proceeding two lemmas, the convergence result of Algorithm 1 can be established.

Theorem 1. *Assume that* $\{(x^s, \lambda^s)\}$ *is the sequence produced by Algorithm 1, where* $x^s = [(x_1^s)^T, (x_2^s)^T, ..., (x_n^s)^T]^T$ *and* $\lambda^s = [\lambda_{e_{ij}}^s]$, $e_{ij} \in E$. *Let* $y^k = \frac{1}{k}\sum_{s=0}^{k-1} x^{s+1}$ *be the ergodic average of* x^s *from step* 1 *to* k. x^* *is the optimal solution of the problem (2.2). Then under Assumption 1, the following relation holds for any* $k \geq 1$ *and for* $0 < \gamma \leq 2$

$$0 \leq f(y^k) - f(x^*) \leq \frac{V^0}{k}. \tag{3.19}$$

 doi:10.20944/preprints202401.2201.v1

where V^0 given in (3.13) is a non-negative term. Furthermore,

$$\lim_{k\to+\infty}\Big(f(y^k)-f(x^*)\Big)=0, \tag{3.20}$$

with the rate of $O(1/k)$.

Proof. It follows from the optimality of x^* that the first inequality in (3.19) is clearly true. Let $x=x^*$ in inequality (3.7), then one has

$$f(x^*)-f(x^{s+1})-(x^*-x^{s+1})^TA^T\lambda^{s+1}+(x^*-x^{s+1})^TQ(x^{s+1}-x^s)-\rho(\gamma-1)(x^*-x^{s+1})^TL_-x^{s+1}\geq 0.$$

Take into consideration that $L_-=A^TA$ and $Ax^*=0$, the above inequality can be rewritten as:

$$f(x^*)-f(x^{s+1})+(x^{s+1})^TA^T\lambda^{s+1}+(x^*-x^{s+1})^TQ(x^{s+1}-x^s)-\rho(1-\gamma)\|Ax^{s+1}\|^2\geq 0.$$

By Lemma 4.2, one has

$$f(x^*)-f(x^{s+1})+V^s\geq V^{s+1}+\frac{\rho\gamma}{2}\|Ax^{s+1}\|^2+\rho(1-\gamma)\|Ax^{s+1}\|^2+\frac{1}{2}\|M(x^{s+1}-x^s)\|^2,$$

and then

$$kf(x^*)-\sum_{s=0}^{k-1}f(x^{s+1})+V^0\geq V^k+\frac{1}{2}\sum_{s=0}^{k-1}\|M(x^{s+1}-x^s)\|^2+\frac{\rho}{2}(2-\gamma)\sum_{s=0}^{k-1}\|Ax^{s+1}\|^2.$$

Due to $V^k\geq 0$ for any k, the following inequality holds for $0<\gamma\leq 2$

$$kf(x^*)-\sum_{s=0}^{k-1}f(x^{s+1})+V^0\geq 0. \tag{3.21}$$

Since the function f is convex, $\sum_{s=0}^{k-1}f(x^{s+1})\geq kf(\frac{1}{k}\sum_{s=0}^{k-1}x^{s+1})$, and then using $y^k=\frac{1}{k}\sum_{s=0}^{k-1}x^{s+1}$, we have

$$kf(x^*)-kf(y^k)+V^0\geq 0,$$

i.e.,

$$f(y^k)-f(x^*)\leq\frac{V^0}{k}. \tag{3.22}$$

Therefore, inequality (3.19) stands. Furthermore, inequality (3.22) implies that

$$\lim_{k\to+\infty}\Big(f(y^k)-f(x^*)\Big)\leq 0.$$

On the other hand, from the optimality of x^*, we have

$$\lim_{k\to+\infty}\Big(f(y^k)-f(x^*)\Big)\geq 0.$$

As a result, $\lim_{k\to+\infty}\Big(f(y^k)-f(x^*)\Big)=0$ and the proof is completed. □

Remark 5. Theorem 1 gives the theoretical upper bound for $f(y^k)-f^*$, which provides the error estimates for the optimal value f^* at each iteration k. The upper bound is consist of two additive items. Both of them approach to zero at the rate $O(1/k)$. In addition, Theorem 1 implies that $f(x^k)$ converges

 doi:10.20944/preprints202401.2201.v1

to the optimal value f^* asymptotically. Furthermore, if at least one function f_i is strongly convex, then the optimal solution x^* is unique, and thus x^k asymptotically approaches to x^*.

Remark 6. When solving the consensus optimization problem (1.1), the convergence condition of Algorithm 1 has less conservative than that in [17], wherein, the convergence of Algorithm 1 can be guaranteed if P_i is symmetric and $P_i \succ \rho d_i I_m$ according to Remark 4.2, while algorithm in [17] requires that P_i is a symmetric positive semi-definite matrix and $P_i \succ (\frac{n}{2-\gamma} - 1)\rho A_i^T A_i = (\frac{n}{2-\gamma} - 1)\rho d_i I_m$ $(0 < \gamma < 2)$.

4. Numerical Experiments

In this section, some numerical experiments are provided to show the validity of Algorithm 1. First, the convergence property of Algorithm 1 is verified. Then the impacts of penalty parameter ρ, damping parameter γ and connectivity ratio d on Algorithm 1 are investigated.

In this section, each edge of the connected network G is generated randomly. The connectivity ratio of the network G is denoted by $d = \frac{2l}{n(n-1)}$. Consider the following consensus optimization problem given in [21]:

$$\min_y \frac{1}{2}\sum_{i=1}^{n}(y-\theta_i)^2, \tag{4.1}$$

where $y \in \mathbb{R}$. Apparently, the optimal solution of this problem is $y^* = \bar{\theta} = \frac{1}{n}\sum_{i=1}^{n}\theta_i$. The consensus optimization problem (4.1) can be reformulated into a distributed version:

$$\begin{aligned} &\min_x f(x) = \frac{1}{2}\sum_{i=1}^{n}(x_i-\theta_i)^2, \\ &\text{subject to } \ x_i = x_j, \ \ \forall (i,j) \in E, \end{aligned} \tag{4.2}$$

where $x = [x_1, x_2, ..., x_n]^T \in \mathbb{R}^n$ and f is convex. Therefore, Algorithm 1 can be used to solve the consensus optimization problem. For the consensus optimization problem (4.2), each θ_i is randomly generated by a normal distribution $N(0,1)$.

The proximal matrix of Algorithm 1 is set by $P_i = \rho d_i I$. In this case, the iteration of x has a closed-form solution, which is shown as follows:

$$x_i^{k+1} = \frac{\rho d_i x_i^k + \rho \sum_{j\in N(i)} x_j^k + \sum_{j\in S(i)} \lambda_{ij}^k - \sum_{j\in P(i)} \lambda_{ji}^k + \theta_i}{1+2\rho d_i},$$

where d_i is the number of neighbors of the agent i.

A. Convergence Property

To illustrate the convergence property of Algorithm 1 for the consensus optimization problem (4.2), ten networks are generated. Each network has $n = 50$ agents and the connectivity ratio of these networks are set as $d = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$ and 1.0, respectively. The algorithm parameters are set as $\rho = 1$ and $\gamma = 1$. The algorithm will be stopped once $\|x^k - x^*\|$ reaches to 10^{-16} or the number of iterations k reaches to 3500, where x^* is the optimal solution of problem (4.1).

Fig. 2 and Fig. 3 respectively depict how the relative error $\frac{\|x^k - x^*\|}{\|x^*\|}$ and constraint violation $\|Ax^k\|$ vary with iteration k. One can find that Algorithm 1 has high accuracy since the relative error can achieve 10^{-13} and the constraint violation can achieve 10^{-16}.

Figure 2. Relative error versus iteration.

Figure 3. Constraint violation versus iteration.

B. Algorithm Parameters ρ and γ

 doi:10.20944/preprints202401.2201.v1

In this part, the impacts of algorithm parameters ρ and γ on the convergence speed of Algorithm 1 are discussed. The networks are generated in the same way as Part A. In order to explore the influences of parameters ρ and γ on Algorithm 1, the convergence speed is denoted by $\xi_{\varepsilon_0} = 1/k_0$, where $\varepsilon_0 > 0$ and k_0 is the number of iterations required to achieve $\|x^{k_0} - x^*\| \leq \varepsilon_0$. Here, the accuracy is set as $\varepsilon_0 = 10^{-6}$.

Choosing damping parameter $\gamma = 1$ and selecting different penalty parameters to solve the problem (4.2), one can get the relationship between the convergence speed ξ_{ε_0} and parameter ρ, which is displayed in Fig. 4. Obviously, if the penalty parameter ρ is too large or too small, the convergence speed of the algorithm is slow. The penalty parameter ρ can be selected from $(0.01, 2)$. In general, a smaller connectivity ratio leads to larger actual optimal parameter ρ^*. As a consequence, when the network is sparse, it is better to select a larger penalty parameter and when it is dense, a smaller penalty parameter will be a nice choice.

Figure 4. Convergence speed versus ρ.

In order to explore the influence of parameter γ on Algorithm 1, the penalty parameter is set as $\rho = 1$ and the damping parameter is set to 60 different values. The numerical results are shown in Fig. 5. Obviously, the convergence speed of Algorithm 1 increases with the damping parameter, and then remains constant. Therefore, $\gamma = 2$ is a great choice.

Figure 5. Convergence speed versus γ.

C. Connectivity Ratio

In this part, the effect of connectivity ratio d on the convergence speed of Algorithm 1 is explored. From Fig. 4, one can find that when penalty parameter ρ takes different values, the impact of connectivity ratio on convergence speed is different. Therefore, the penalty parameter is set to six different values $\rho = 0.005, 0.01, 0.05, 0.1, 1$ and 2, respectively.

We generate 30 networks with $n = 50$ agents, whose connectivity ratio are set to 30 different values : $\frac{1}{30}$, $\frac{2}{30}$, ..., 1. From Fig. 6, one can find that when the penalty parameter takes a smaller value, such as $\rho = 0.005, 0.01$ or 0.05, the convergence speed of Algorithm 1 generally slows down with the increase of connectivity ratio, and the opposite is true when the penalty argument takes a bigger value, such as $\rho = 0.1, 1$ or 2 from Fig. 7. It is worth noting that when the network is very sparse, for example $d = 0.05$, no matter what the penalty parameter value is, the convergence speed is slow. Therefore, on the premise of ensuring network connectivity, few edges can be added to increase information exchange between agents.

Figure 6. Convergence speed versus d.

Figure 7. Convergence speed versus d.

5. Application to A Logistic Regression Problem

In this section, the proposed distributed Jacobi-proximal ADMM algorithm is applied to a logistic regression problem, which is a widely used machine learning model[22,23].

The network $G = \{V, E\}$ is generated with $n = 50$ agents. The connectivity ratio is set as $d = 0.3$ and the edges are generated randomly. The network generated is given in Fig. 8. Each agent has n_i training samples, which denoted by $\{\mathbf{w}_{ij}, y_{ij}\}_{j=1}^{n_i}$, where $\mathbf{w}_{ij} \in \mathbb{R}^p$ and $y_{ij} \in \{1, -1\}$.

Figure 8. The network of problem (5.1).

 doi:10.20944/preprints202401.2201.v1

The distributed logistic regression problem is described as follows:

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|^2 + \frac{1}{N}\sum_{i=1}^{n}\sum_{j=1}^{n_i} log(1 + e^{-y_{ij}\mathbf{w}_{ij}^T\mathbf{x}}), \tag{5.1}$$

where $N = \sum_{i=1}^{n} n_i$ is the total number of samples. The dimension of feature is set as $p = 3$, the number of samples n_i is generated by a uniform distribution $U(1, 20)$, and the parameter $\mathbf{w}_{ij}$ is generated by a normal distribution $N(0, 1)$. The generation rule of the label y_{ij} is shown as follows:

$$y_{ij} = \begin{cases} 1, & \text{if } u_{ij} \geq 0.5, \\ -1, & \text{if } u_{ij} < 0.5, \end{cases}$$

where u_{ij} is generated by a uniform distribution $U(0, 1)$.

The distributed logistic regression problem (5.1) can be formulated as

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x}_i), \\ &\text{subject to } \; x_i = x_j, \;\; \forall (i, j) \in E, \end{aligned} \tag{5.2}$$

where $x = [x_1^T, x_2^T, ..., x_n^T]^T$ and $f_i(\mathbf{x}_i) = \frac{1}{2n}\|\mathbf{x}_i\|^2 + \frac{1}{N}\sum_{j=1}^{n_i} log(1 + e^{-y_{ij}\mathbf{w}_{ij}^T\mathbf{x}_i})$. Obviously, problem (5.2) can be solved by Algorithm 1.

The convergence path of Algorithm 1 is compared with the Jocobi-Proximal ADMM (JP-ADMM) algorithm in [17]. To investigate the performances of the two algorithms, the penalty parameter is set to ρ =0.01, 0.1 and 1, respectively. In addition, the damping parameter is set to two different values $\gamma = 1$ and $\frac{3}{2}$. The proximal matrix of Algorithm 1 and algorithm in [17] are set as $P_i = \rho d_i I$ and $P_i = [(\frac{n}{2-\gamma} - 1)\rho d_i + 1]I$, respectively. Every algorithm is stopped once $\|x^k - x^{k-1}\|$ reaches to 10^{-5} or the number of iterations k reaches to 1000. One can find that the convergence speed of Algorithm 1 is significantly faster than that in [17] from Fig. 9 and Fig. 10.

Figure 9. Objective value f^k ($\gamma = \frac{1}{2}$).

Figure 10. Objective value f^k ($\gamma = \frac{3}{2}$).

6. Conclusions

In this paper, a distributed ADMM algorithm is put forward to solve a consensus convex optimization problem over a connected network. The proposed algorithm is a Jacobi-proximal ADMM algorithm and the proximal matrix is smaller than existing algorithms. The convergence of the algorithm is proved and its convergence rate is $O(1/k)$. Extensive numerical experiments are provided to verify the convergence of the algorithm. Besides, the impacts of penalty parameter, damping parameter and connectivity ratio on the proposed algorithm are investigated. Finally, an application of the proposed algorithm to a logistic regression problem is implemented and its performance is compared with that of another ADMM algorithm in [17].

Acknowledgments: This research was supported by the National Natural Science Foundation of China (Grant number: 11801051) and the Natural Science Foundation of Chongqing (Grant number: cstc2019jcyj-msxmX0075).

References

1. Y.L. Pan, Distributed optimization and statistical learning for large-scale penalized expectile regression, J. Korean Stat. Soc. 50 (2021) 290-314.
2. G. Chen, J.Y. Li, A fully distributed ADMM-based dispatch approach for virtual power plant problems, Appl. Math. Model. 58 (2018) 300-312.

3. G. Droge, H. Kawashima, M.B. Egerstedt, Continuous-time proportional-integral distributed optimisation for networked systems, J. Control Decis. 1 (2014) 191-213.
4. B. Gharesifard, J. Cortés, Continuous-time distributed convex optimization on weight-balanced digraphs, IEEE Trans. Autom. Control 59 (2014) 781-786.
5. Y.N. Zhu, W.W. Yu, G.H. Wen, G.R. Chen, W. Ren, Continuous-time distributed subgradient algorithm for convex optimization with general constraints, IEEE Trans. Autom. Control 64 (2019) 1694-1701.
6. W. Zhu, H.B. Tian, Distributed convex optimization via proportional-integral-differential algorithm, Meas. Control 55 (2021) 13-20.
7. A. Nedic, A. Ozdaglar, Distributed subgradient methods for multi-agent optimization, IEEE Trans. Autom. Control 54 (2009) 48-61.
8. C. Xi, U.A. Khan, Distributed subgradient projection algorithm over directed graphs, IEEE Trans. Autom. Control 62 (2017) 3986-3992.
9. S. Liu, Z.R. Zhang, L.H. Xie, Convergence rate analysis of distributed optimization with projected subgradient algorithm, Automatic 83 (2017) 162-169.
10. D. Jakovetić, J. Xavier, J.M.F. Moura, Cooperative convex optimization in networked systems: augmented Lagrangian algorithms with directed gossip communication, IEEE Trans. Signal Process. 59 (2011) 3889-3902.
11. S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends Mach. Learn. 3 (2010) 1-122.
12. R. Zhang, J.T. Kwok, Asynchronous distributed ADMM for consensus optimization, in: Proceedings of the 31st International Conference on Machine Learning, 2014, pp. 3689-3697.
13. E. Wei, A. Ozdaglar, Distributed alternating direction method of multipliers, in: Proceedings of the IEEE Conference on Decision and Control, 2012, pp. 5445-5450.
14. J.Q. Yan, F.H. Guo, C.Y. Wen, G.Q. Li, Parallel alternating direction method of multipliers, Inf. Sci. 507 (2020) 185-196.
15. W. Shi, Q. Ling, K. Yuan, G. Wu, W. Yin, On the linear convergence of the ADMM in decentralized consensus optimization, IEEE Trans. Signal Process. 62 (2014) 1750-1761.
16. M. Hong, H. Davood, M. Zhao, Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 2402-2433.
17. W. Deng, M.J. Lai, Z.M. Peng, W.T. Yin, Parallel multi-block ADMM with $o(1/k)$ convergence, J. Sci. Comput. 71 (2017) 712-736.
18. W. Deng, W.T. Yin, On the global and linear convergence of the generalized alternating direction method of multipliers, J. Sci. Comput. 66 (2016) 889-916.
19. M. Sun, H.C. Sun, Improved proximal ADMM with partially parallel splitting for multi-block separable convex programming, Appl. Math. Comput. 58 (2018) 151-181.
20. M. Fazel, T.K. Pong, D.F. Sun, P. Tseng, Hankel matrix rank minimization with applications to system identification and realization, SIAM J. Matrix Anal. Appl. 34 (2013) 946-977.
21. M. Rabbat, R. Nowak, Distributed optimization in sensor networks, in: Proceedings of the third International Symposium on Information Processing in Sensor Networks, 2004, pp. 20-27.
22. L.J. Wang, M. Guo, K. Sawada, J. Lin, J.C. Zhang, A comparative study of landslide susceptibility maps using logistic regression, frequency ratio, decision tree, weights of evidence and artificial neural network, Geosci. J. 20 (2016) 117-236.
23. B.Y. Kim, S.J. Shin, Principal weighted logistic regression for sufficient dimension reduction in binary classification, J. Korean Stat. Soc. 48 (2019) 194-206.