Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 1007032, 14 pages
https://doi.org/10.1155/2022/1007032

Research Article

A Distributed Online Newton Step Algorithm for Multi-Agent Systems

Xiaofei Chu

School of Business, Henan University of Science and Technology, Luoyang 471023, China

Correspondence should be addressed to Xiaofei Chu; chuxf701015@sina.com

Received 12 July 2022; Accepted 11 October 2022; Published 28 October 2022

Academic Editor: Binchang Wang

Most of the current algorithms for solving distributed online optimization problems are based on the first-order method, which are simple in computation but slow in convergence. Newton's algorithm with fast convergence speed needs to calculate the Hessian matrix and its inverse, leading to computationally complex. A distributed online optimization algorithm based on Newton's step is proposed in this paper, which constructs a positive definite matrix by using the first-order information of the objective function to replace the inverse of the Hessian matrix in Newton's method. The convergence of the algorithm is proved theoretically and the regret bound of the algorithm is obtained. Finally, numerical experiments are used to verify the feasibility and efficiency of the proposed algorithm. The experimental results show that the proposed algorithm has an efficient performance on practical problems, compared to several existing gradient descent algorithms.

1. Introduction

In recent years, with the development of computer network technology, distributed optimization algorithms [1–3] have been successfully applied to solve large-scale optimization problems, which are considered to be an effective method. Distributed optimization decomposes a large-scale optimization problem into multiple subproblems according to different agents in a multi-agent network. Different agents calculate their associated subproblems simultaneously and communicate information with their immediate neighbor agents. And, all the agents finally obtain a common optimal solution that can minimize the sum of their objective functions through the exchange of information along with their respective optimization iteration. Many problems in science and engineering can be modeled as problems, such as machine learning [4], signal tracking and location [5], sensor networks [6, 7], and smart grids [8].

Distributed optimization assumes that the local objective function is known and invariant. However, many practical problems in diverse fields are affected by their environment and the corresponding objective function is changing all the time, which requires the optimization process of these problems in the online setting. In the distributed online optimization problem, each agent has a limited amount of information. Only when one agent makes a decision with the current information can it know the relevant information of its corresponding objective function, which leads to the inevitable outcome: the decision it makes is not the optimal and the difference, so-called regret, exists between make-decisions of all agents, respectively. Regret is one of the most straightforward measures of the performance of an online algorithm. Obviously, the smaller the regret, the better the performance of the algorithm. Since the implementation of the algorithm is completed after multiple iterations, we theoretically require that the regret generated by multiple iterations should gradually approach to zero along with the increasing of iterative number. That is, if the corresponding cumulative regret bound of the algorithm can be got, the regret bound should be sublinear convergence with respect to the number of iteration.

Distributed online optimization and its applications in multi-agent systems have become a hot research area nowadays in the systems and control community. Inspired by the distributed dual average algorithm in [3], the authors in [9] proposed a distributed online weighted dual average

algorithm for the distributed optimization problem on dynamic networks and obtained a square-root regret bound. Yan et al. [10] introduced a distributed online projected subgradient descent algorithm and achieved a square-root regret for convex locally cost functions and a logarithmic regret for strongly convex locally cost function. In [11], a distributed stochastic subgradient online optimization algorithm is proposed. In the case that the local objective function is convex and strongly convex, the convergence of the algorithm is proved, and the corresponding regret bounds are obtained respectively. For more references on distributed online algorithms, see [12–21].

The distributed online optimization algorithm based on the first-order information is simple in a calculation but converges slowly in most cases. In the traditional optimization method, Newton's method converges faster than the first-order information when it uses the second-order gradient information of the objective function. Some scholars have applied it to distributed optimization problems [22–24] to improve the convergence of distributed online optimization algorithm. However, these algorithms need to compute and store the Hessian matrix of the objective along with its inverse at each iteration, which will inevitably affect the efficiency. To overcome such inconvenience, inspired by the algorithm mentioned in reference [25], we propose a distributed online Newton step algorithm, which can achieve the effect of the Newton method by using the first information of the objective function.

The major contributions of this article are as follows: (i) for the distributed online optimization problem, we propose a distributed online Newton step algorithm which can not only avoid the deficiencies of calculation and storage in the process of Newton method implementation but also achieve the effectiveness of Newton's method. In the algorithm, the first-order gradient of the object function has been used to construct a positive definite matrix, which is similar to the inverse of the Hessian matrix in the Newton method. Moreover, the convergence of the proposed algorithm is proved theoretically, and the regret bound of the algorithm is obtained. (ii) We apply the proposed algorithm to a practical problem. The effectiveness and practicality of the algorithm is verified by numerical experiments. Meanwhile, we compare the proposed algorithm with several existing gradient descent algorithms, and the results show that the convergence rate of this algorithm is obviously faster than the gradient descent algorithms.

The rest of this paper is organized as follows: in Section 2, we discuss some closely related works about distributed Newton method. Some necessary mathematical preliminaries and assumptions which will be used in this paper are introduced in Section 3. Our algorithm is stated in Section 4, and the concrete proof of the convergence of the algorithm is presented in Section 5. The performance of the proposed algorithm is verified in comparison with several gradient descent algorithms over the practical problem in Section 6, and then the conclusion of this paper is included in Section 7.

2. Related Work

Newton and Quasi-Newton methods are recognized as a class of effective algorithms in solving optimization problems. The iterative formula of Newton method is as follows:

$$x(k+1) = x(k) - \alpha(k) H(k)^{-1} g(k), \tag{1}$$

where $H(k)^{-1}$ is the inverse of Hessian matrix $\nabla^2 f(x(k))$. Newton's method needs to calculate the second derivative of the objective function and the Hessian matrix obtained may not be positive definite. In order to overcome these shortcomings, some scholars put forward the Quasi-Newton method. The basic idea of the Quasi-Newton method is to structure an approximate matrix without the second derivative of the objective function to instead of the inverse of the Hessian matrix in the Newton method. Different methods of constructing approximate matrix represent different Quasi-Newtonian methods.

Although the Quasi-Newton method is recognized as a more efficient algorithm, it is seldom used in a distributed environment because the distribution approximation of Newton steps is difficult to design. Mark Eisen et al. [26] proposed a decentralized Quasi-Newton Method. In this method, the idea of determining the Newton direction is to utilize the inverse of the Hessian matrix to approximate the curvature of the cost function of the agent and its neighbor agents. This method has good convergence but faces storage and computational deficiencies for large data sets (The approach involves the power of matrices of sizes $np \times np$ with n being the total number of nodes and p being the number of features). Aryan Mokhtari, Qing Ling, and Alejandro Ribeiro in [27] proposed a Network Newton method. In this method, a distributed computing matrix is constructed by equivalent transformation of the original problem, which can replace the original Hessian matrix, so as to realize the distributed solution of the problem. The authors proved that the algorithm can converge to the approximate value of the optimal parameter at least at the linear rate. They further demonstrated the convergence rate of the algorithm and analyzed several practical implementation matters in the literature [28]. Rasul Tutunov et al. [29] proposed a distributed Newton optimization algorithm based on consensus. By using the sparsity of the dual Hessian matrix, they reconstructed the calculation of Newton steps into the method of solving diagonally dominant linear equations and realized the distributed calculation of Newton's method. They also theoretically proved that the algorithm has superlinear convergence similar to centralized Newton's algorithm in the field of the optimal solution. Although these algorithms realize the distributed computation of the Newton method, they need to calculate the inverse of the Hessian, which is expensive for the algorithm.

Motivated by these observations, for the online setting, we propose a distributed Newton-step algorithm which can achieve a convergence rate approximate to Newton's method on the basis of distributed computing, and the inverse of the approximate Hessian matrix can be easily calculated. Numerical experimental results show that our

algorithm can run significantly faster than the algorithms in [9, 10, 19] with a lower computational cost in preiteration.

3. Preliminaries

In this section, some notational conventions and basic notions are introduced first. Then, we provide a brief description distributed online optimization problem. At the same time, some concepts will be used and relevant assumptions are represented in this paper.

3.1. Some Conceptions and Assumptions. The n-dimension Euclidean space is denoted by $\mathbb{R}^n$, $\mathcal{X}$ is a subset of $\mathbb{R}^n$, and $\|\cdot\|$ represents the Euclidean norm. Strongly convex functions are defined as follows:

Definition 1. [30] Let $f(x)$ be a differentiable function on $\mathbb{R}^n$, $\nabla f(x)$ be the gradient of function $f(x)$ at x, and $\mathcal{X} \in \mathbb{R}^n$ be a convex subset. Then $f(x)$ is strictly convex on $\mathcal{X}$ if and only if

$$f(x) > f(x_0) + \langle \nabla f(x_0), x - x_0 \rangle, \tag{2}$$

for all $(x_0, x) \in \mathcal{X} \times \mathcal{X}$.

Lemma 1 (see [25]). *Function $f(x)$: $\mathcal{X} \longrightarrow \mathbb{R}$ is differentiable on the set $\mathcal{X}$ with a diameter D, and $\alpha > 0$ is a constant. For $\forall x \in \mathcal{X}, \|\nabla f(x)\| \le L$ and $\exp(-\alpha f(x))$ is concave, then when $\beta \le \min\{1/4LD, \alpha\}$, for any $x, y \in \mathcal{X}$, the following inequation holds:*

$$f(x) \ge f(y) + \nabla f(y)^T (x - y) + \frac{\beta}{2}(x - y)^T \nabla f(y) \nabla f(y)^T (x - y). \tag{3}$$

Some notations about matrices are given to be used in our proof of the convergence of the algorithm. Denote the space of all $n \times n$ matrices by $\mathbb{R}^{n \times n}$. For a matrix $A = (a_{ij})_{n \times n} \in \mathbb{R}^{n \times n}$, a_{ij} represents the entry of A at the ith row and the jth column. $A^T = (a_{ji})_{n \times n}$ is the transpose of A. $|A|$ denotes the determinant of A, and λ_i is the ith eigenvalue of the matrix A. Then, the next equations are set up: $|A| = \prod_{i=1}^{n} \lambda_i$, $\mathbf{tr}A = \sum_{i=1}^{n} a_{ii} = \sum_{i=1}^{n} \lambda_i$. In addition, for any vector $x, y, z \in \mathbb{R}^n$, equation $x^T A z + y^T A z = (x + y)^T A z$ is set up.

Definition 2. [31] Matrix $A \in \mathbb{R}^{n \times n}$ is positive definite, if and only if for any $x \in \mathbb{R}^n$ and $x \neq \mathbf{0}$ ($\mathbf{0}$ denotes an n-dimensional vector where all the entries are 0), $x^T A x > 0$.

3.2. Distributed Online Optimization Problem. We consider a multiagent network system with multiple agents, each agent i is associated with a strictly convex function (with bounded first and second derivatives) $f_{t,i}(x)$: $\mathbb{R}^n \longrightarrow \mathbb{R}$, and the function $f_{t,i}(x)$ is varying over time. All the agents cooperate to solve the following general convex consensus problem:

$$\min \sum_{i=1}^{n} f_{t,i}(x), \quad \text{subject to } x \in \mathcal{X}. \tag{4}$$

At each round $t = 1, \ldots, T$, the ith agent is required to generate a decision point $x_i(t) \in \mathcal{X}$ according to its current local information as well as the information received from its immediate neighbors. Then, the adversary responds to each $agent's$ decision with a loss function $f_{t,i}(x)$: $\mathcal{X} \longrightarrow \mathbb{R}$ and each agent gets the loss $f_{t,i}(x_i(t))$. The communication between agents is specified by an undirected graph $G = (V, E)$, where $V = \{1, \ldots, n\}$ is a vertex set, and $E \subset V \times V$ denotes an edge set. Undirected means if $(i, j) \in E$ then $(j, i) \in E$. Each agent i can only communicate directly with its immediate neighbors $N(i) = j \in V | (i, j) \in E$. The goal of the agents is to seek a sequence of decision points $x_i(t) \in \mathcal{X}, i \in V$, so that the regret with respect to each agent i regarding any fixed decision $x^* \in \mathcal{X}$ in hindsight

$$R_T(x_i(t), x) = \sum_{t=1}^{T} \sum_{i=1}^{n} \left(f_{t,i}(x_i(t)) - f_{t,i}(x^*) \right), \tag{5}$$

is sublinear in T.

Throughout this paper, we make the following assumptions:

(i) each cost function $f_{t,i}(x)$ is strictly convex and twice continuous differentiable and L-Lipschitz on the convex set $\mathcal{X}$

(ii) $\mathcal{X}$ is compact and the Euclidean diameter of $\mathcal{X}$ is bounded by D

(iii) $\exp(-\alpha f_{ti}(x))$ is concave in the set $\mathcal{X}$ for all t and i

By assumption (i), the function $f_{ti}(x)$ is convex in the set $\mathcal{X}$, and with some reasonable assumptions over the domains of the value of α and x, $\exp(-\alpha f_{ti}(x))$ is concave in the set $\mathcal{X}$. In addition, the Lipschitz condition (i) implies that for any $x \in \mathcal{X}$ and any gradient g_i, we have the following equation:

$$\|g_i\| \le L. \tag{6}$$

4. Distributed Online Newton Step Algorithm

For problem (4), we assume that information can be exchanged among each agent in a timely manner, that is, the network topology graph between n agents is a complete graph. The communication between agents in our algorithm is modeled by a doubly stochastic symmetric P, so that $1 > p_{ij} > 0$ only if $(i, j) \in E$, else $p_{ij} = 0$, and $\sum_{j=1}^{n} p_{ij} = \sum_{j \in N(i)} p_{ij} = 1$ for all $i \in V$, $\sum_{i=1}^{n} p_{ij} = \sum_{i \in N(j)} p_{ij} = 1$ for all $j \in V$.

4.1. Algorithm. The distributed online Newton step algorithm is presented in Algorithm 1.

The projection function used in this algorithm is defined as follows:

$$\prod_{\mathcal{X}}^{A}(y) = \arg\min_{x \in \mathcal{X}} (y - x)^T A (y - x), \tag{7}$$

where A is a positive definite matrix.

4.2. Algorithm Analysis. In this algorithm, when a decision $x_i(t)$ is made by the agent i with the current information, the corresponding cost function $f_{t,i}(x)$ can be obtained. So we can get the gradient $g_i(t) = \nabla f_{t,i}(x_i(t))$. Construct a symmetric positive definite matrix $H_i(t) = \sum_{r=1}^{t} g_i(r) g_i(r)^T + \epsilon I_n$, then the direction of iteration is constructed by utilizing $H_i(t)^{-1}$ which always exists to replace the inverse of the Hessian matrix in Newton's method. Take the linear combination of the current iteration point of agent i and the current iteration point of its neighbor agent as the starting point of the new iteration along with the size $1/\beta$, and the projection operation is used to get the next iteration point $x_i(t+1)$.

The calculation of $H_i(t)$ and its inverse $H_i(t)^{-1}$ in the algorithm can be seen from Step 7, $H_i(t) = \sum_{r=1}^{t} g_i(r) g_i(r)^T + \epsilon I_n = \sum_{r=1}^{t-1} g_i(r) g_i(r)^T + \epsilon I_n + g_i(t) g_i(t)^T = H_i(t-1) + g_i(t) g_i(t)^T$, which shows that $H_i(t)$ can be computed via using the previous approximation matrix $H_i(t-1)$ as well as the gradient $g_i(t)$ at step t. Therefore, we do not have to store all the gradients from the previous t-step iteration, at the same time, as shown by the following equation (32):

$$\left(A + uv^T\right)^{-1} = A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1 + v^T A^{-1} u}. \tag{8}$$

Let $H_i(t-1) = A, u = v = g_i(t)$, and $H_i(0) = \epsilon I_n$, then $H_i(0)^{-1} = 1/\epsilon I_n$, the inverse of $H_i(t)$ can be got simply. It is the same thing as solving for $H_i(t)$, we just use the information from the current and the previous step.

5. Convergence Analysis

Now, the main result of this paper is stated in the following.

Theorem 1. *Give the sequence of $x_i(t)$ and $z_i(t)$ generated by Algorithm 1 for all $i \in V$, $x^* = \arg\min_{x \in \mathcal{X}} \sum_{i=1}^{n} f_{t,i}(x)$, and the regret with respect to agent i's action is*

$$R_T(x^*, x_i(t)) = \sum_{t=1}^{T} \left(f_{ti}(x_i(t)) - f_{ti}(x^*)\right) \leq \frac{\epsilon D^2 \beta}{2} + \frac{n}{2\beta} \ln\left(\frac{TL^2}{\epsilon} + 1\right) + C + C_3 \ln\left(\frac{T\widehat{L}^2 + \epsilon}{\widehat{L}^2 + \epsilon}\right), \tag{9}$$

where $C = C_1 + C_2$, $C_1 = \beta\eta^2 n^2 D^2 (1 - 2ln\eta)L^2/(ln\eta)^2 - 1/ln\eta)$, $C_2 = n^3 DL(M+m)^2 \eta / Mm\epsilon(1-)\eta^2$, $C_3 = n^4 L^2 (M+m)^4 / 8\beta M^2 m^2 (1-\eta)^2 \widehat{L}^2$, $\eta = \max_{1 \leq i,j \leq n}\{p_{ij}\}$, *and* M, m, $\widehat{L}$*are constants, and* $\widehat{L} \leq 1/t \sum_{r=1}^{t} \|g_i(r)\|^2$.

From Theorem 1, the regret bound of Algorithm 1 is sublinear convergence with respect to iterative number T, that is, $\lim_{T \longrightarrow \infty} R_T(x^*, x_i(t))/T = 0$. Note that, the regret bound is related to the scale of the network. Specifically, as the network grows in size, the regret bound value also increases. In addition, the value of the regret bound is also influenced by the values of parameter ϵ and the diameter of the convex set $\mathcal{X}$. The value of η indirectly reflects the connectivity of the network implying that the smaller the value of η , the smaller the regret bound of the algorithm.

To prove the conclusion of Theorem 1, we first give some lemmas and their proofs.

Lemma 2. *For any fixed $i \in V$, let $f_{ti}(x) = f_t(x)$, then $\nabla f_{ti}(x_i(t)) = \nabla f_t(x_i(t)) = g_i(t)$, and the following bound holds for any $j \in V$ and $x \in \mathcal{X}$*

$$\sum_{t=1}^{T} \left(f_t(x_i(t)) - f_t(x^*)\right) \leq \sum_{t=1}^{T} g_i(t)^T \left(x_i(t) - x^*\right) - \frac{\beta}{2} \sum_{t=1}^{T} \left(x_i(t) - x^*\right)^T g_i(t) g_i(t)^T \left(x_i(t) - x^*\right), \tag{10}$$

where $\beta = 1/2 \min\{1/4LD, \alpha\}$.

Proof. According to the assumption that the function $f_t(x)$ is strictly convex and continuous differentiable in convex set $\mathcal{X}$, and $x_i(t) \in \mathcal{X}, x^* \in \mathcal{X}$, by Lemma 1 we can obtain

$$f_t(x_i(t)) - f_t(x^*) \leq g_i(t)^T \left(x_i(t) - x^*\right) - \frac{\beta}{2} \left(x_i(t) - x^*\right)^T g_i(t) g_i(t)^T \left(x_i(t) - x^*\right). \tag{11}$$

Summing up over $t = \{1, 2, \ldots, T\}$ can get the conclusion of Lemma 2.

From Lemma 2, if the upper bound of the right side of the inequality can be obtained, the upper bound of the left side can be obtained, too. Therefore, we are committed to solving the upper bound of the right side of the above equation. □

Lemma 3. *Let $y_i(t) = \sum_{j=1}^{n} p_{ij} x_j(t) - x_i(t)$, and the following bound holds for any $j \in V$ and any $x \in \mathcal{X}$,*

$$\begin{aligned}
&\sum_{t=1}^{T} g_i(t)^T \left(x_i(t) - x^*\right) - \frac{\beta}{2} \sum_{t=1}^{T} \left[x_i(t) - x^*\right]^T g_i(t) g_i(t)^T \left[x_i(t) - x^*\right] \\
\leq &\frac{\beta}{2} \epsilon D^2 - \sum_{t=1}^{T} g_i(t)^T y_i(t) + \beta \sum_{t=1}^{T} y_i(t)^T H_i(t) \left(x_i(t) - x^*\right) \\
&+ \frac{\beta}{2} \sum_{t=1}^{T} y_i(t)^T H_i(t) y_i(t) + \frac{1}{2\beta} \sum_{t=1}^{T} g_i(t)^T H_i(t)^{-1} g_i(t).
\end{aligned} \tag{12}$$

Proof. according to Algorithm 1, we have the following equation:

$$z_i(t+1) = \sum_{j=1}^{n} p_{ij} x_j(t) - \frac{1}{\beta} H_i(t)^{-1} g_i(t), \tag{13}$$

so

$$\begin{aligned}
z_i(t+1) - x^* &= \sum_{j=1}^{n} p_{ij} x_j(t) - \frac{1}{\beta} H_i(t)^{-1} g_i(t) - x^* \\
&= x_i(t) - x^* + \sum_{j=1}^{n} p_{ij} x_j(t) - x_i(t) - \frac{1}{\beta} H_i(t)^{-1} g_i(t) \\
&= x_i(t) - x^* + y_i(t) - \frac{1}{\beta} H_i(t)^{-1} g_i(t),
\end{aligned} \tag{14}$$

then we can obtain the following next equation:

$$\begin{aligned}
&\left(z_i(t+1) - x^*\right)^T H_i(t) \left(z_i(t+1) - x^*\right), \\
&= \left[x_i(t) - x^*\right]^T H_i(t) \left[x_i(t) - x^*\right] + \left[y_i(t) - \frac{1}{\beta} H_i(t)^{-1} g_i(t)\right]^T H_i(t) \left(x_i(t) - x^*\right) \\
&\quad + \left(x_i(t) - x^*\right)^T H_i(t) \left[y_i(t) - \frac{1}{\beta} H_i(t)^{-1} g_i(t)\right] \\
&\quad + \left[y_i(t) - \frac{1}{\beta} H_i(t)^{-1} g_i(t)\right]^T H_i(t) \left[y_i(t) - \frac{1}{\beta} H_i(t)^{-1} g_i(t)\right] \\
&= \left[x_i(t) - x^*\right]^T H_i(t) \left[x_i(t) - x^*\right] + 2 y_i(t)^T H_i(t) \left(x_i(t) - x^*\right) \\
&\quad - \frac{2}{\beta} g_i(t)^T \left(x_i(t) - x^*\right) - \frac{2}{\beta} g_i(t)^T y_i(t) + y_i(t)^T H_i(t) y_i(t) + \frac{1}{\beta^2} g_i(t)^T H_i(t)^{-1} g_i(t).
\end{aligned} \tag{15}$$

Since $x_i(t+1)$ is the projection of $z_i(t+1)$ in the norm induced by $H_i(t)$, it is a well known fact that (see [25] section 3.5 Lemma 3.9)

$$\left[x_i(t+1) - x^*\right]^T H_i(t) \left[x_i(t+1) - x^*\right] \leq \left[z_i(t+1) - x^*\right]^T H_i(t) \left[z_i(t+1) - x^*\right]. \tag{16}$$

This fact together with (15) gives

$$\begin{aligned}&[x_i(t+1)-x^*]^T H_i(t)[x_i(t+1)-x^*]\\ &\le [x_i(t)-x^*]^T H_i(t)[x_i(t)-x^*]+2y_i(t)^T H_i(t)(x_i(t)-x^*)\\ &\quad -\frac{2}{\beta}g_i(t)^T(x_i(t)-x^*)-\frac{2}{\beta}g_i(t)^T y_i(t)\\ &\quad +y_i(t)^T H_i(t)y_i(t)+\frac{1}{\beta^2}g_i(t)^T H_i(t)^{-1}g_i(t).\end{aligned} \tag{17}$$

Summing both sides of (17) from $t = 1$ to T, we obtain the following equation:

$$\begin{aligned}&\sum_{t=1}^{T}[x_i(t+1)-x^*]^T H_i(t)[x_i(t+1)-x^*]\\ &\le \sum_{t=1}^{T}[x_i(t)-x^*]^T H_i(t)[x_i(t)-x^*]\\ &\quad -\frac{2}{\beta}\sum_{t=1}^{T}g_i(t)^T y_i(t)-\frac{2}{\beta}\sum_{t=1}^{T}g_i(t)^T(x_i(t)-x^*)\\ &\quad +2\sum_{t=1}^{T}y_i(t)^T H_i(t)(x_i(t)-x^*)+\sum_{t=1}^{T}y_i(t)^T H_i(t)y_i(t)\\ &\quad +\frac{1}{\beta^2}\sum_{t=1}^{T}g_i(t)^T H_i(t)^{-1}g_i(t),\end{aligned} \tag{18}$$

that is

$$\begin{aligned}&-\sum_{t=1}^{T}[x_i(t)-x^*]^T(H_i(t+1)-H_i(t))[x_i(t)-x^*]\\ &\le [x_i(1)-x^*]^T\left(H_i(1)-g_i(1)g_i(1)^T\right)[x_i(1)-x^*]\\ &\quad -\frac{2}{\beta}\sum_{t=1}^{T}g_i(t)^T y_i(t)-\frac{2}{\beta}\sum_{t=1}^{T}g_i(t)^T(x_i(t)-x^*)\\ &\quad -[x_i(T+1)-x^*]^T H_i(T)[x_i(T+1)-x^*]\\ &\quad +\sum_{t=1}^{T}y_i(t)^T H_i(t)y_i(t)+2\sum_{t=1}^{T}y_i(t)^T H_i(t)(x_i(t)-x^*)\\ &\quad +\frac{1}{\beta^2}\sum_{t=1}^{T}g_i(t)^T H_i(t)^{-1}g_i(t).\end{aligned} \tag{19}$$

According to Algorithm 1, $H_i(t+1)-H_i(t)=g_i(t)g_i(t)^T$, then

$$\begin{aligned}&\sum_{t=1}^{T}[x_i(t)-x^*]^T(H_i(t+1)-H_i(t))[x_i(t)-x^*]\\ &\quad =\sum_{t=1}^{T}[x_i(t)-x^*]^T g_i(t)g_i(t)^T[x_i(t)-x^*],\end{aligned} \tag{20}$$

thus we obtain

$$\begin{aligned}&\sum_{t=1}^{T}g_i(t)^T(x_i(t)-x^*)-\frac{\beta}{2}\sum_{t=1}^{T}[x_i(t)-x^*]^T g_i(t)g_i(t)^T[x_i(t)-x^*]\\ &\le \frac{\beta}{2}[x_i(1)-x^*]^T\left(H_i(1)-g_i(1)g_i(1)^T\right)[x_i(1)-x^*]\\ &\quad -\sum_{t=1}^{T}g_i(t)^T y_i(t)+\beta\sum_{t=1}^{T}y_i(t)^T H_i(t)(x_i(t)-x^*)\\ &\quad -\frac{\beta}{2}[x_i(T+1)-x^*]^T H_i(T)[x_i(T+1)-x^*]\\ &\quad +\frac{\beta}{2}\sum_{t=1}^{T}y_i(t)^T H_i(t)y_i(t)+\frac{1}{2\beta}\sum_{t=1}^{T}g_i(t)^T H_i(t)^{-1}g_i(t).\end{aligned} \tag{21}$$

Due to $H_i(1)-g_i(1)g_i(1)^T=\epsilon I_n$, then

$$\begin{aligned}&[x_i(1)-x^*]^T\left(H_i(1)-g_i(1)g_i(1)^T\right)[x_i(1)-x^*],\\ &=\epsilon[x_i(1)-x^*]^T[x_i(1)-x^*]=\epsilon\|x_i(1)-x^*\|^2\le\epsilon D^2.\end{aligned} \tag{22}$$

And, since $H_i(T)$ is positive definite, and $\beta>0$, so $-\beta/2[x_i(T+1)-x^*]^T H_i(T)[x_i(T+1)-x^*]\le 0$. Combining (21) and (22), we can state

$$\begin{aligned}&\sum_{t=1}^{T}g_i(t)^T(x_i(t)-x^*)-\frac{\beta}{2}\sum_{t=1}^{T}[x_i(t)-x^*]^T g_i(t)g_i(t)^T[x_i(t)-x^*]\\ &\le \frac{\beta}{2}\epsilon D^2-\sum_{t=1}^{T}y_i(t)^T g_i(t)+\beta\sum_{t=1}^{T}y_i(t)^T H_i(t)(x_i(t)-x^*)\\ &\quad +\frac{\beta}{2}\sum_{t=1}^{T}y_i(t)^T H_i(t)y_i(t)+\frac{1}{2\beta}\sum_{t=1}^{T}g_i(t)^T H_i(t)^{-1}g_i(t).\end{aligned} \tag{23}$$

Thus, the proof of Lemma 3 is completed.
Next, we consider the last term of (23). □

Lemma 4. *For any $i \in V$, we can get the following bound holding:*

$$\frac{1}{2\beta}\sum_{t=1}^{T}g_i(t)^T H_i(t)^{-1}g_i(t)\le\frac{n}{2\beta}\log\left(\frac{TL^2}{\epsilon}+1\right). \tag{24}$$

Proof. Note that,

(1) Input: convex set χ, maximum round number T.
(2) $\beta = 1/2 \min\{1/4LD, \alpha\}$.
(3) Initialize: $x_i(1) \in \chi$, $\forall i \in V$.
(4) **for** $t = 1, \cdots\cdots, T$ **do**
(5) The adversary reveals $f_{t,i}, \forall i \in V$.
(6) Compute gradients $g_i(t) \in \partial f_{t,i}(x_i(t))$, $\forall i \in V$.
(7) Compute $H_i(t) = \sum_{r=1}^{t} g_i(r) g_i(r)^T + \epsilon I_n$, $\forall i \in V$.
(8) **for** each $i \in V$ **do**
(9) $z_i(t+1) = \sum_{j=1}^{n} P_{ij} x_j(t) - 1/\beta H_i(t)^{-1} g_i(t)$
(10) $x_i(t+1) = \prod_{\chi}^{H_i(t)} (z_i(t+1))$
(11) **end for**
(12) **end for**

Algorithm 1: The Distributed Online Newton Step Algorithm (D-ONS).

$$g_i(t)^T H_i(t)^{-1} g_i(t) = H_i(t)^{-1} \bullet g_i(t) g_i(t)^T = H_i(t)^{-1} \bullet \left(H_i(t) - H_i(t-1)\right), \tag{25}$$

where for matrices $A, B \in \mathbb{R}^{n \times n}$, we denote by $A \bullet B = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{ij}$ the inner product of these matrices as vectors in $\mathbb{R}^{n^2}$. For real numbers $a, b \in \mathbb{R}^+$ and the logarithm function $y = \ln x$, the Taylor expansion of y in a is $y = \log x = \log a + 1/a\,(x - a) + R_n(x)$. So $\log b \le \log a + 1/a\,(b - a)$, implying $a^{-1}(a - b) \le \log a/b$. An analogous fact holds for the positive definite matrices, i.e., $A^{-1} \bullet (A - B) \le \log |A| / |B|$, where $|A|, |B|$ denote the determinant of the matrix A, B (see the detailed proof in [25]). This fact gives us (for convenience we denote $H_i(0) = \epsilon I_n$)

$$\begin{aligned} \sum_{t=1}^{T} g_i(t)^T H_i(t)^{-1} g_i(t) &= \sum_{t=1}^{T} H_i(t)^{-1} \bullet \left(H_i(t) - H_i(t-1)\right), \\ &\le \sum_{t=1}^{T} \log \frac{|H_i(t)|}{|H_i(t-1)|} = \log \frac{|H_i(T)|}{|H_i(0)|}. \end{aligned} \tag{26}$$

Since $H_i(T) = \sum_{t=1}^{T} g_i(t) g_i(t)^T + \epsilon I_n$ and $\|g_i(t)\| \le L$, from the properties of matrices and determinants, we know that the largest eigenvalue of $H_i(T)$ is $TL^2 + \epsilon$ at most. Hence $|H_i(T)| \le (TL^2 + \epsilon)^n$ and $|H_i(0)| = \epsilon^n$, then

$$\sum_{t=1}^{T} g_i(t)^T H_i(t)^{-1} g_i(t) \le \log \frac{|H_i(T)|}{|H_i(0)|} \le n \log\left(\frac{TL^2}{\epsilon} + 1\right). \tag{27}$$

Combining the above factors, we obtain the following equation:

$$\frac{1}{2\beta} \sum_{t=1}^{T} g_i(t)^T H_i(t)^{-1} g_i(t) \le \frac{n}{2\beta} \log\left(\frac{TL^2}{\epsilon} + 1\right). \tag{28}$$

The proof of Lemma 4 is completed.

According to Algorithm 1, $z_i(t+1) = \sum_{j=1}^{n} p_{ij} x_j(t) - 1/\beta H_i(t)^{-1} g_i(t)$, where $-H_i(t)^{-1} g_i(t)$ is the direction of iteration. Using the knowledge of matrix analysis, we have the following conclusions. □

Lemma 5. *For any $i \in V, 1 \le t \le T$,*

$$\left\|H_i(t)^{-1} g_i(t)\right\| \le \frac{(M+m)^2 n^2 L}{4Mm\left(\sum_{r=1}^{t} \left\|g_i(r)\right\|^2 + \epsilon\right)}, \tag{29}$$

where $m = \min\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, $M = \max\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, $0 < m \le M$, and $\lambda_i (i = 1, \ldots, n)$ is the i th eigenvalue of $H_i(t)$.

This conclusion gives us that when the number of iterations increases, $\|H_i(t)^{-1} g_i(t)\|$ converges to zero, which ensures the consistency of the algorithm. The detailed proof can be seen in Appendix A.

Now, we consider the norm of vector $y_i(t)$, $z_i(t+1) - x^*$ and get the following inequation.

Lemma 6. *For any $1 \le i \le n, 1 \le t \le T$, let $\eta = \max_{1 \le i,j \le n} p_{ij}$, $0 < \eta < 1$, then*

$$\left\|z_i(t+1) - x^*\right\| \le nD\eta^t + \frac{(M+m)^2 n^2 L}{4\beta Mm\left(\sum_{r=1}^{t} \left\|g_i(r)\right\|^2 + \epsilon\right)(1-\eta)}, \tag{30}$$

and

$$\left\|y_i(t)\right\| \le 2\left(nD\eta^t + \frac{(M+m)^2 n^2 L}{4\beta Mm\left(\sum_{r=1}^{t} \left\|g_i(r)\right\|^2 + \epsilon\right)(1-\eta)}\right), \tag{31}$$

where n is the size of the network, T is the total number of iterations. The specific proof is represented in Appendix B.

Next, we turn our attention to the bound of the following term $\beta \sum_{t=1}^{T} y_i(t)^T H_i(t) (x_i(t) - x^*) - \sum_{t=1}^{T} y_i(t)^T g_i(t) + \beta/2 \sum_{t=1}^{T} y_i(t)^T H_i(t) y_i(t)$. By combining the knowledge of vectors and matrices, we get Lemma 7.

Lemma 7. *For any $i \in V$, the following inequality holds*

$$\beta\sum_{t=1}^{T} y_i(t)^T H_i(t)\left(x_i(t)-x^*\right)-\sum_{t=1}^{T} y_i(t)^T g_i(t)\frac{\beta}{2}\sum_{t=1}^{T} y_i(t)^T H_i(t) y_i(t) \le 2\beta\sum_{t=1}^{T}\left(\sum_{r=1}^{t}\left\|g_i(r)\right\|^2+\epsilon\right)\left(nD+\frac{(M+m)^2 n^2 L}{4\beta Mm\left(\sum_{r=1}^{t}\left\|g_i(r)\right\|^2+\epsilon\right)(1-\eta)}\right)^2, \le C+C_3\ln\left(\frac{T\hat{L}^2+\epsilon}{\hat{L}^2+\epsilon}\right), \tag{32}$$

where $C_1=\beta\eta^2 n^2 D^2\left((1-2ln\eta)L^2/(ln\eta)^2-1/ln\eta\right)$, $C_2=n^3 DL(M+m)^2\eta/Mm\epsilon(1-\eta)^2$, $C_3=n^4 L^2(M+m)^4/8\beta M^2 m^2(1-\eta)^2\hat{L}^2$, *and* $C=C_1+C_2$.

Proof. According to Algorithm 1, we can state

$$z_i(t+1)=\sum_{j=1}^{n} p_{ij}x_j(t)-\frac{1}{\beta}H_i(t)^{-1}g_i(t), \tag{33}$$

where β is a positive constant, and $H_i(t)^{-1}$ is the inverse of matrix $H_i(t)$. We obtain the following equation:

$$g_i(t)=\beta H_i(t)\left(\sum_{j=1}^{n} p_{ij}x_j(t)-z_i(t+1)\right). \tag{34}$$

Now, by multiplying both sides of this equation by the vector $y_i(t)^T$, we can obtain the following equation:

$$y_i(t)^T g_i(t)=\beta y_i(t)^T H_i(t)\left(\sum_{j=1}^{n} p_{ij}x_j(t)-z_i(t+1)\right), \tag{35}$$

then the left of (32) can be written as follows:

$$\begin{aligned}&\sum_{t=1}^{T} y_i(t)^T H_i(t)\left(x_i(t)-x^*\right)+\frac{\beta}{2}\sum_{t=1}^{T} y_i(t)^T H_i(t) y_i(t)-\sum_{t=1}^{T} y_i(t)^T g_i(t)\\ &=\beta\sum_{t=1}^{T} y_i(t)^T H_i(t)\left(x_i(t)-x^*\right)+\frac{\beta}{2}\sum_{t=1}^{T} y_i(t)^T H_i(t) y_i(t)\\ &\quad-\beta\sum_{t=1}^{T} y_i(t)^T H_i(t)\left[\sum_{j=1}^{n} p_{ij}x_j(t)-z_i(t+1)\right].\end{aligned} \tag{36}$$

The matrix $H_i(t)$ is symmetric and positive definite, which means that $y_i(t)^T H_i(t) y_i(t)\ge 0$, therefore we can obtain the following equation:

$$\begin{aligned}&\beta\sum_{t=1}^{T} y_i(t)^T H_i(t)\left(x_i(t)-x^*\right)+\frac{\beta}{2}\sum_{t=1}^{T} y_i(t)^T H_i(t) y_i(t)\\ &\quad-\beta\sum_{t=1}^{T}\left[\sum_{j=1}^{n} p_{ij}x_j(t)-z_i(t+1)\right]^T H_i(t) y_i(t)\\ &\le\beta\sum_{t=1}^{T} y_i(t)^T H_i(t)\left(x_i(t)-x^*\right)+\beta\sum_{t=1}^{T} y_i(t)^T H_i(t) y_i(t)\\ &\quad-\beta\sum_{t=1}^{T} y_i(t)^T H_i(t)\left[\sum_{j=1}^{n} p_{ij}x_j(t)-z_i(t+1)\right]\\ &\le\beta\sum_{t=1}^{T} y_i(t)^T H_i(t)\left(z_i(t+1)-x^*\right)\le\beta\sum_{t=1}^{T}\left|y_i(t)^T H_i(t)\left(z_i(t+1)-x^*\right)\right|\\ &\le\beta\sum_{t=1}^{T}\sqrt{\left\|y_i(t)\right\|_{H_i(t)}^2\left\|z_i(t+1)-x^*\right\|_{H_i(t)}^2}.\end{aligned} \tag{37}$$

Here, we apply the Cauchy–Schwarz inequation: $|x^T A y|^2 \le \|x\|_A^2 \|y\|_A^2$, where A is a $n \times n$ Hermite matrix and A is positive semidefinite.

Next, we consider the super bound of $\|y_i(t)\|_{H_i(t)}^2$ and $\|z_i(t+1) - x^*\|_{H_i(t)}^2$. According to the Step 7 of Algorithm 1, $H_i(t) = \sum_{r=1}^{t} g_i(r) g_i(r)^T + \epsilon I_n = H_i(t-1) + g_i(t) g_i(t)^T$, so

$$\begin{aligned}
\|y_i(t)\|_{H_i(t)}^2 &= y_i(t)^T H_i(t) y_i(t) = y_i(t)^T \left(H_i(t-1) + g_i(t) g_i(t)^T\right) y_i(t), \\
&= \|y_i(t)\|_{H_i(t-1)} + \|y_i(t)^T g_i(t)\|^2 \le \|y_i(t)\|_{H_i(t-1)} + \|y_i(t)\|^2 \|g_i(t)\|^2, \\
&\le \|y_i(t)\|_{H_i(0)} + \|y_i(t)\|^2 \|g_i(1)\|^2 + \cdots + \|y_i(t)\|^2 \|g_i(t)\|^2, \\
&\le \epsilon \|y_i(t)\|^2 + \sum_{r=1}^{t} \|y_i(t)\|^2 \|g_i(r)\|^2 \le \left(\epsilon + \sum_{r=1}^{t} \|g_i(r)\|^2\right) \|y_i(t)\|^2.
\end{aligned} \tag{38}$$

Similarly, we have the following equation:

$$\|z_i(t+1) - x^*\|_{H_i(t)}^2 \le \left(\epsilon + \sum_{r=1}^{t} \|g_i(r)\|^2\right) \|z_i(t+1) - x^*\|^2. \tag{39}$$

Combining the results of Lemmas 5 and 6, we have the following equation:

$$\|y_i(t)\|_{H_i(t)}^2 \|z_i(t+1) - x^*\|_{H_i(t)}^2 \le 4 \left(\epsilon + \sum_{r=1}^{t} \|g_i(r)\|^2\right)^2 \left(n D \eta^t + \frac{(M+m)^2 n^2 L}{4\beta M m \left(\sum_{r=1}^{t} \|g_i(r)\|^2 + \epsilon\right)(1-\eta)}\right)^4, \tag{40}$$

then

$$\begin{aligned}
&\beta \sum_{t=1}^{T} y_i(t)^T H_i(t) \left(x_i(t) - x^*\right) + \frac{\beta}{2} \sum_{t=1}^{T} y_i(t)^T H_i(t) y_i(t) \\
&\quad - \beta \sum_{t=1}^{T} \left[\sum_{j=1}^{n} p_{ij} x_j(t) - z_i(t+1)\right]^T H_i(t) y_i(t) \\
&\le \beta \sum_{t=1}^{T} \left(y_i(t)\right)^T H_i(t) \left(z_i(t+1) - x^*\right) \\
&\le \beta \sum_{t=1}^{T} \sqrt{\|y_i(t)\|_{H_i(t)}^2 \|z_i(t+1) - x^*\|_{H_i(t)}^2}, \\
&\le 2\beta \sum_{t=1}^{T} \left(\epsilon + \sum_{r=1}^{t} \|g_i(r)\|^2\right) \left(n D \eta^t + \frac{(M+m)^2 n^2 L}{4\beta M m \left(\sum_{r=1}^{t} \|g_i(r)\|^2 + \epsilon\right)(1-\eta)}\right)^2.
\end{aligned} \tag{41}$$

Thus, we complete the proof of Lemma 7.

Putting all these together, Theorem 1 can be proved as follows. □

Proof of Theorem 1. According to the assumptions, $f(x)$ is strictly convex, and the function $\exp(-\alpha f(x))$ is concave in $\mathcal{X}$ when the value of α is sufficiently small. Setting $\beta = \min\{1/4LD, \alpha\}$, combined with axioms 2–7, we can obtain the regret bound

$$\begin{aligned} R_T(x^*, x_i(t)) &= \sum_{t=1}^{T} \left(f_{ti}(x_i(t)) f_{ti}(x^*)\right) \\ &\leq \frac{\epsilon D^2 \beta}{2} + \frac{n}{2\beta}\log\left(\frac{TL^2}{\epsilon} + 1\right) + C + C_3 \log\left(\frac{T\hat{L}^2 + \epsilon}{\hat{L}^2 + \epsilon}\right). \end{aligned} \tag{42}$$

The values of the parameters in equation (15) are the same as Theorem 1. □

6. Numerical Experiment

In order to verify the performance of our proposed algorithm, we conducted a numerical experiment on an online estimation over a distributed sensor network which is mentioned in reference [9]. In a distributed sensor network, there are n sensors (See Figure 1 in [9]). Each sensor is connected to one or more sensors. It is assumed that each sensor is connected to a processing unit. Finally, the processing units are integrated to obtain the best evaluation of the environment. The specific model is as follows: given a closed convex set $\mathcal{X} = \{x \in \mathbb{R}^n | \|x\|_2 \leq x_{\max}\}$, the observation vector $y_{t,i}$: $\mathbb{R}^n \longrightarrow \mathbb{R}^d$ represents the i th sensor measurement at time t which is uncertain and time-varying due to the sensor's susceptibility to unknown environmental factors such as jamming. The sensor is assumed (not necessarily accurately) to have a linear model of the form $h_i(x) = H_i x$, where $H_i \in \mathbb{R}^{n\times d}$ is the observation matrix of sensor i and $\|H_i\|_1 \leq h_{\max}$ for all i. The objective is to find the argument $\hat{x} \in \mathcal{X}$ that minimizes the cost function $f(\hat{x}) = 1/n \sum_{i=1}^{n} f_{t,i}(\hat{x})$, namely,

$$\begin{aligned} &\min \frac{1}{n}\sum_{i=1}^{n} f_{t,i}(\hat{x}), \\ &\text{subject to } \hat{x} \in \mathcal{X}, \end{aligned} \tag{43}$$

where the cost function associated with sensor i is $f_{t,i}(\hat{x}) = 1/2\|y_{t,i}(x) - H_i\hat{x}\|_2^2$. Since the observed value $y_{t,i}$ changes with time t, only when we calculate the value of $\hat{x}_i(t)$ can we get the local error of the i th sensor at time t. In other words, due to modeling errors and uncertainties in the environment, the local error functions are allowed to change over time.

In an offline setting, each sensor i has a noisy observation $y_{t,i} = H_i x + v_{t,i} = H_i x + v_{1,i}$ for all $t = 1, 2, \ldots, T$, where $v_{t,i}$ is generally assumed to be (independent) white noise at time t. In this case, the centralized optimal estimate for problem (37) is

$$\hat{x}^* = \left(\sum_{i=1}^{n} H_i^T H_i\right)^{-1} \sum_{i=1}^{n} H_i^T y_{1,i}(x). \tag{44}$$

While in an online setting, the white noise $v_{t,i}$ varies with time t (see [9]). We assume $v_{t,i} \sim \mathbf{U}(-1/4, 1/4)$ (The white noise $v_{t,i}$ is uniformly distributed on the interval $(-1/4, 1/4)$). In the proposed distributed online algorithm, each sensor i calculate $\hat{x}_i(t) \in \mathcal{X}$ based on the local information available to it and then an "oracle" reveals the cost $f_{t,i}(\hat{x})$ at time step t.

The performance of the proposed algorithm is discussed based on the following aspects:

6.1. The Analysis of the Algorithm Performance. The numerical experiments consist of two parts: the impact of network size on the performance of the D-ONS and the effect of network connectivity on the effectiveness of the algorithm iterations.

We carried out numerical experiments at $n = 1$, $n = 2$ and $n = 100$, respectively. Figure 1 depicts the convergence curves of the algorithm for different network sizes. According to Figure 1, it is obvious that the average regret decreases fast and the algorithm can converge on different scaled networks as the number of the agent in the network increase. Especially, when $n = 1$, the problem is equivalent to a centralized optimization problem, and our distributed optimization algorithm can reach the same effect as the centralized algorithm.

According to Theorem 1, the effectiveness of the algorithm is directly affected by the connectivity of the network, so we verify the algorithm under different network topology. (i) Complete graph. All the agents are connected to each other. (ii) Cycle graph. Each agent is only connected to its two immediate neighbors. (iii) Watts–Strogatz. The connectivity of random graphs is related to the average degree and connection probability. Here, let the average degree of the graph is 3 and the probability of connection is 0.6. As shown in Figure 2, D-ONS can lead to a significantly faster convergence on a complete graph than a cycle graph and has the similar convergence on Watts–Strogatz. The experimental result is consistent with the theoretical analysis results in this paper.

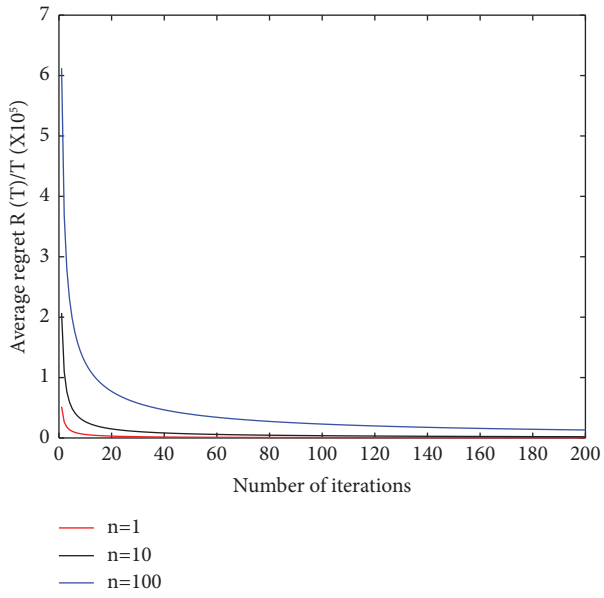

Figure 1: Comparison $n = 1$, $n = 10$, $n = 100$.

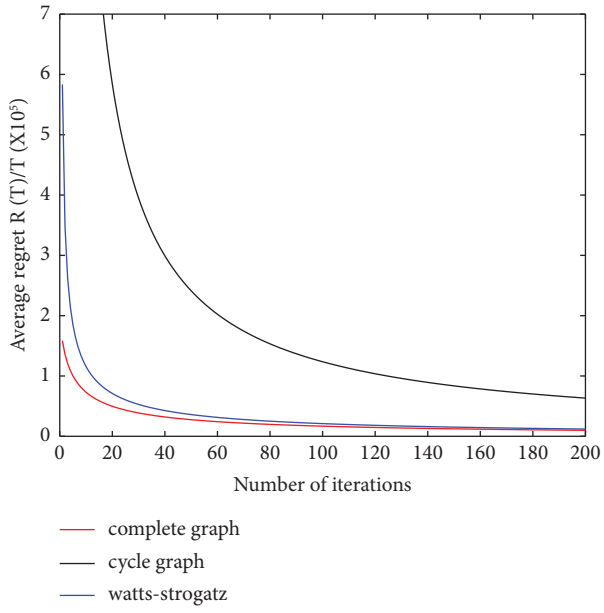

Figure 2: Comparison under different topology.

6.2. Performance Comparison of Algorithms. To verify the performance of the proposed algorithm, we compared the proposed algorithm with the class algorithms D-OGD in [10], D-ODA in [9] and the algorithm in [19]. The parameters in these algorithms are based on their theoretical proofs. The network topology relationship among agents is complete, and the size of the network is the same ($n = 10$). As shown in Figure 3, the presented algorithm D-ONS displays better performance with faster convergence and higher accuracy than D-ODA, D-OGD, and the algorithm in [19].

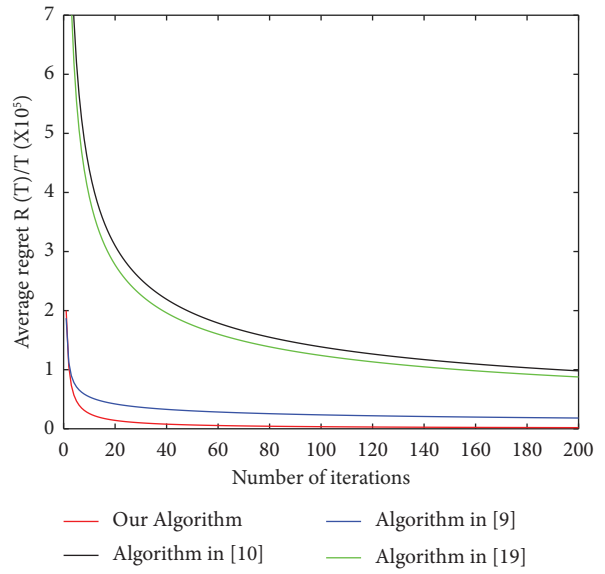

Figure 3: The convergence curves of compared algorithms.

7. Conclusion and Discussion

A distributed online optimization algorithm based on Newton step is proposed for a multiagent distributed online optimization problem, where the local objective function is strictly convex and quadratic continuously differentiable. In each iteration, the gradient of the current iteration point is used to construct a positive definite matrix, and then the direction of the next iteration is constructed by substituting this positive matrix for the inverse of the Hessian matrix in Newton's method. Through theoretical analysis, the regret bound of the algorithm is obtained, and the regret bound converges sublinear with respect to the number of iterations. Numerical examples also demonstrate the feasibility and effectiveness of the proposed algorithm. Simulation results indicate significant convergence rate improvement of our algorithm relative to the existing distributed online algorithms based on first-order methods.

Appendix

A. The Proof of Lemma 5

Proof. First, we consider the value of $\|H_i(t)^{-1} g_i(t)\|^2$. Let $\lambda_i(t)$ be the *ith* eigenvalue of $H_i(t)$, then $1/\lambda_i(t)$ is the *ith* eigenvalue of $H_i(t)^{-1}$. Applying the Schweitzer inequation (See 2.11 in [33]), we can get

$$\mathbf{tr}\left(H_i(t)^{-1}\right) \leq \frac{(M+m)^2 n^2}{4Mm\mathbf{tr}\left(H_i(t)\right)}, \tag{A.1}$$

where $0 < m \leq \lambda_i(t) \leq M$ for $i = 1, 2, \cdots\cdots, n$. Obviously, $H_i(t)$ is symmetric positive definite implying that $H_i(t)^{-1}$ is symmetric positive definite, and $(H_i(t)^{-1})^T = H_i(t)^{-1}$. By the definition of vector's norm, we have

$$\begin{aligned}&\left\|H_i(t)^{-1}g_i(t)\right\|^2\\&=\mathbf{tr}\left(H_i(t)^{-1}\right)^2\left(g_i(t)g_i(t)^T\right)\\&\le\left(\mathbf{tr}\left(H_i(t)^{-1}\right)\right)^2\mathbf{tr}\left(g_i(t)g_i(t)^T\right)\le\left(\frac{(M+m)^2n^2}{4Mm\mathbf{tr}\left(H_i(t)\right)}\right)^2L^2.\end{aligned}\tag{A.2}$$

For any $x\in\mathbb{R}^n$, $x^T(g_i(t)g_i(t)^T)x=x^Tg_i(t)g_i(t)^Tx=\|x^Tg_i(t)\|\ge0$, $g_i(t)g_i(t)^T$ is positive semidefinite, and $\mathbf{tr}(g_i(t)g_i(t)^T)=\|g_i(t)\|^2\le L^2$. Due to $\mathbf{tr}(H_i(t))=\sum_{r=1}^t\|g_i(r)\|^2+n\epsilon$, we have

$$\left\|H_i(t)^{-1}g_i(t)\right\|\le\frac{(M+m)^2n^2L}{4Mm\left(\sum_{r=1}^t\left\|g_i(r)\right\|^2+n\epsilon\right)}\le\frac{(M+m)^2n^2L}{4Mm\left(\sum_{r=1}^t\left\|g_i(r)\right\|^2+\epsilon\right)}.\tag{A.3}$$

The proof of Lemma 5 is completed. □

B. Proof of Lemma 5

Proof. Consider $z_i(t+1)=\sum_{j=1}^n p_{ij}x_j(t)-1/\beta H_i(t)^{-1}g_i(t)$ and $\sum_{j=1}^n p_{ij}=1$, thus

$$\begin{aligned}&\left\|z_i(t+1)-x^*\right\|=\left\|\sum_{j=1}^n p_{ij}x_j(t)-\frac{1}{\beta}H_i(t)^{-1}g_i(t)-x^*\right\|,\\&=\left\|\sum_{j=1}^n p_{ij}\left(x_j(t)-x^*\right)-\frac{1}{\beta}H_i(t)^{-1}g_i(t)\right\|\\&\le\sum_{j=1}^n p_{ij}\left\|z_j(t)-x^*\right\|+\frac{1}{\beta}\left\|H_i(t)^{-1}g_i(t)\right\|\\&=\sum_{j=1}^n p_{ij}\left\|\sum_{j=1}^n p_{ij}x_j(t-1)-\frac{1}{\beta}H_j(t-1)^{-1}g_j(t-1)-x^*\right\|+\frac{1}{\beta}\left\|H_i(t)^{-1}g_i(t)\right\|\\&\le\sum_{j=1}^n p_{ij}\left\|\sum_{j=1}^n p_{ij}x_j(t-1)-x^*\right\|+\frac{1}{\beta}\sum_{j=1}^n p_{ij}\left\|H_j(t-1)^{-1}g_j(t-1)\right\|\\&\quad+\frac{1}{\beta}\left\|H_i(t)^{-1}g_i(t)\right\|\ \le\cdots\cdots\ \le\left\|\sum_{j=1}^n p_{ij}^tx_j(1)-x^*\right\|+\frac{1}{\beta}\left\|H_i(t)^{-1}g_i(t)\right\|\\&\quad+\frac{1}{\beta}\left\|\sum_{k=1}^{t-1}\sum_{j=1}^n p_{ij}^{t-k}H_j(k)^{-1}g_j(k))\right\|,\\&\le\sum_{j=1}^n p_{ij}^t\left\|x_j(1)-x^*\right\|+\frac{1}{\beta}\left\|H_i(t)^{-1}g_i(t)\right\|+\frac{1}{\beta}\sum_{k=1}^{t-1}\sum_{j=1}^n p_{ij}^{t-k}\left\|H_j(k)^{-1}g_j(k)\right\|\Big),\\&\le nD\eta^t+\frac{(M+m)^2n^2L}{4Mm\beta\left(\sum_{r=1}^t\left\|g_i(r)\right\|^2+\epsilon\right)(1-\eta)},\end{aligned}\tag{A.4}$$

where $x_j(t)$ is the projections of $z_j(t)$ onto the convex set $\mathcal{X}$. Similarly, we can get

$$\left\|y_i(t)\right\| \leq 2\left(nD\eta^t + \frac{(M+m)^2 n^2 L}{4Mm\beta\left(\sum_{r=1}^{t}\left\|g_i(r)\right\|^2 + \epsilon\right)(1-\eta)}\right). \tag{A.5}$$

The proof of Lemma 6 is completed. □

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The author thanked Guanglei Sun, PhD and Jia Liu, Master from Henan University of Science and Technology for the arrangement of the article and the improvement of the content. The authors also thanked the referees for their valuable comments on the original manuscript. This work was supported in part by the National Natural Science Foundation of China under Grant nos. 11471102 and 12071112; in part, by the basic research projects in the University of Henan Province under Grant no. 20ZX001.

References

[1] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, New Jersey, USA, 1989.

[2] A. Nedi, A. Ozdaglar, Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[3] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: convergence analysis and network scaling," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, 2012.

[4] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[5] V. Lesser and C. Ortiz, "Distributed Sensor Networks: A Multiagent Perspective," in *Kluwer Academic Publishers*, M. Tambe, Ed., vol. 9, 2003.

[6] M. Rabbat and R. Nowak, "Distributed Optimization in Sensor Networks," in *Proceedings of the the 3rd International Symposium On Information Processing In Sensor Networks*, pp. 20–27, Berkeley, CA, USA, April 2004.

[7] D. Li, K. Wong, Y. Hu, and A. Sayeed, "Detection, Classification and Tracking of Targets in Distributed Sensor Networks," *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 17–29, 2002.

[8] D. K. Molzahn, F. Dorfler, H. Sandberg et al., "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2941–2962, 2017.

[9] S. Hosseini, A. Chapman, and M. Mesbahi, "Online distributed convex optimization on dynamic networks," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3545–3550, 2016.

[10] F. Yan, S. Sundaram, S. V. N. Vishwanathan, and Y. Qi, "Distributed autonomous online learning: regrets and intrinsic privacy-preserving properties," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2483–2493, 2013.

[11] J. Zhu, C. Xu, J. Guan, and D. O. Wu, "Differentially private distributed online algorithms over time-varying directed networks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 4–17, 2018.

[12] H. Gokcesu and S. S. Kozat, "Efficient online convex optimization with adaptively minimax optimal dynamic regret," Available: arXiv: 1907.00497, 2019.

[13] D. Mateosnunez and J. Cortes, "Distributed online convex optimization over jointly connected digraphs," *IEEE Transactions on Network Science and Engineering*, vol. 1, no. 1, pp. 23–37, 2014.

[14] M. Raginsky, N. Kiarashi, and R. Willett, "Decentralized Online Convex Programming with Local Information," in *Proceedings of the American Control Conference*, pp. 5363–5369, San Francisco, CA, USA, July 2011.

[15] M. Akbari, B. Gharesifard, and T. Linder, "Distributed online convex optimization on time-varying directed graphs," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 3, pp. 417–428, 2017.

[16] D. Yuan, D. W. C. Ho, and G. P. Jiang, "An adaptive primal-dual subgradient algorithm for online distributed constrained optimization," *IEEE Transactions on Cybernetics*, vol. 48, no. 11, pp. 3045–3055, 2018.

[17] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, 2014.

[18] C. Xu, J. Zhu, and D. O. Wu, "Decentralized online learning methods based on weight-balancing over time-varying digraphs," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 3, pp. 394–406, 2021.

[19] X. Yi, X. Li, L. Xie, and H. J. Karl, "Distributed online convex optimization with time-varying coupled inequality constraints," [Online]. Available: arXiv: 1903.04277, 2019.

[20] X. Yi, X. Li, L. Xie, and K. H. Johansson, "Distributed online convex optimization with time-varying coupled inequality Constraints," *IEEE Transactions on Signal Processing*, vol. 68, pp. 731–746, 2020.

[21] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, "A decentralized second-order method with exact linear convergence rate for consensus optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 507–522, 2016.

[22] D. Bajovic, D. Jakovetic, N. Krejic, and N. K. Jerinkic, "Newton-like method with diagonal correction for distributed optimization," *SIAM Journal on Optimization*, vol. 27, no. 2, pp. 1171–1203, 2017.

[23] E. Wei, A. Ozdaglar, and A. Jadbabaie, "A distributed Newton method for network utility maximization," *Decision and Control (CDC),2010 49th IEEE Conference on*, vol. 58, pp. 1816–1821, 2010.

[24] M. Eisen, A. Mokhtari, and A. Ribeiro, "An Asynchronous Quasi-newton Method for Consensus Optimization," Signal And Information Processing (GlobalSIP)," in *Proceedings of the 2016 IEEE Global Conference*, pp. 570–574, Washington, DC, USA, December 2016.

[25] E. Hazan, *Efficient Algorithms for Online Convex Optimization and Their Applications*, Princeton University, USA, 2006.

[26] M. Eisen, A. Mokhtari, and A. Ribeiro, "Decentralized Quasi-Newton Methods," *IEEE transactions on signal processing*, vol. 65, no. 10, 2016.
[27] A. Mokhtari, Q. Ling, and A. Ribeiro, "Newton Newton Part I: Algorithm and Convergence," 2015, https://arxiv.org/abs/1504.06017.
[28] A. Mokhtari, Q. Ling, and A. Ribeiro, "Newton Newton Part II: Convergence Rate and Implementation," 2015, https://arxiv.org/abs/1504.06020.
[29] R. Tutunov, H. B. Ammar, and A. Jadbabaie, "A Distributed Newton Method for Large Scale Consensus Optimization," 2016, https://arxiv.org/abs/1606.06593.
[30] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.
[31] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, USA, 1985.
[32] M. S. Bartlett, "An inverse matrix adjustment arising in discriminant analysis," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 107–111, 1951.
[33] D. S. Mitrinovi, *Analytic Inequalities*, Springer Berlin Heidelberg, Berlin, Germany, 1970.