Inference Attack in Distributed Optimization via Interpolation and Manipulation

Jisheng Xu, Zhiyu He, Chongrong Fang, Jianping He, and Yunfeng Peng

***Abstract*— We study the problem of inference attack in distributed optimization, with adversarial agents aiming to obtain the sensitive information of some critical agent in a network. Different from existing privacy-preserving and resilient distributed optimization algorithms, we propose inference algorithms from the perspective of launching well-designed attacks to help infer sensitive local information. The key idea is that by utilizing the critical agent's neighborhood information and the predefined update protocol, adversarial agents can not only interpolate the gradient of its local objective function, but also manipulate it to converge to its own local minimizer. The proposed algorithms can thus obtain approximations of the gradient or the minimizer of the local objective of this critical agent. We characterize the performance through interpolation errors, as well as distances to the optimal value and optimal point of the local objective. Numerical simulations are presented to verify the effectiveness of these algorithms.**

I. Introduction

Distributed optimization has attracted considerable interests recently [1]. Its main emphasis is on exploiting local computations and communication to achieve collaborative optimization of a global objective. However, the interaction between neighboring agents can also cause the risk of disclosing sensitive local information. Moreover, the overall performance may be severely affected if there exist adversarial attacks against participating agents. These issues motivate the growing researches on privacy preservation and resilience in distributed optimization.

Privacy preservation aims at preventing sensitive local states, objectives, or constraints from being easily disclosed. It has been investigated that obfuscating local information with well-designed random noises is effective for protecting sensitive information of agents in a network [2]. In this regard, issues like achieving differential privacy [3] and quantifying the relationship between estimation accuracy and disclosure probability [4] are explored. The above studies mainly focus on reducing data utility in face of honest-but-curious [5] or external evasdroppers. The premise is that the predefined protocols are not violated during execution.

Resilience against attack consists in guranteeing the performance of cooperative optimization in the presence of adversarial agents that deviate from the specified protocols. To be concrete, some malicious agents may send extreme data to mislead other agents to some unwanted points [6]. To resist such adverse effects, various algorithms have been proposed. Mean-Subsequence Reduced algorithm [6] enables normal nodes to cooperatively optimize the sum or the average of their local objective functions in spite of the existence of some adversarial agents. Other algorithms like [7]–[9] can detect the faults and isolate them based on some global knowledge of the network. Existing resilient algorithms focus on keeping the effectiveness of the network rather than on privacy preservation in terms of attacks.

There have also been works that adopt the angle of attackers and design manipulation mechanisms. For instance, Ding *et al.* [10] and Wu *et al.* [11] propose attack algorithms to steer the final state to an arbitrarily selected point by sending fixed gradient estimations or states. However, an unexplored perspective is how the adversarial agents can systematically launch attacks to help effectively infer sensitive local information. Consider a certain critical agent in the network. This agent updates its states based on its local information and the information received from its neighbors. In this case, its malicious neighbors may exploit a finite sample of such transmitted information to approximately infer sensitive local objective functions. Moreover, through interacting with its neighbors, the critical agent may be manipulated to expose its local information actively and unconsciously.

Motivated by the aforementioned observations, we design a novel neighborhood-data-based inference attack algorithm, where adversarial agents desire to obtain sensitive information from some critical neighbor agent in the network. According to the amount of available information for attackers, two scenarios are considered, i.e., attack with *full* or *partial* information on the neighborhood, respectively. To highlight central ideas, in this paper we investigate a simplified setup of inference attack in the context of the classical distributed gradient descent algorithm [12]. This investigation can provide useful insights to launch similar attacks against state-of-the-art privacy-preserving distributed optimization algorithms.

Specifically, we first observe the states of the critical agent and its neighbors, infer the gradient value of this critical agent at several points, and obtain an approximation of the gradient of its local function via polynomial interpolation. Then, we alter the data sent from the adversarial agents to the critical agent, affect the neighborhood data used by the critical agent, and manipulate it into converging to its local minimizer unwittingly, so that the critical agent gradually broadcasts its sensitive information to all its neighbors.

It is worth mentioning that because the data sent by

The authors are with the Department of Automation, Shanghai Jiao Tong University, Key Laboratory of System Control and Information Processing, Ministry of Education of China, and Shanghai Engineering Research Center of Intelligent Control and Management, Shanghai 200240, China. Emails: {Jimmy_xu,hzy970920,crfang,jphe,pengyf}@sjtu.edu.cn.
This work was supported in part by the NSF of China under Grants 62072308, 62103266 and 61973218.

the attackers to other agents is calculated according to the normal protocol, most agents cannot identify the attackers. In the full-information scenario, we establish the error bound for the interpolation of the gradient of the sensitive local objective, and indicate the optimal point of the network in the presence of the proposed algorithm. In the partial-information scenario, the upper bound of the estimation error of our manipulation-based attack algorithm is explicitly characterized, and the convergence rate of the attack is given. The main contributions are summarized as follows.

- We investigate the problem where adversarial agents systematically launch attacks to infer the sensitive local information of some critical agent in a distributed optimization network.
- We propose interpolation-based and manipulation-based attack algorithms to obtain an approximation of the gradient and infer the minimizer of the local objective of the critical agent, respectively.
- We characterize the performance of the proposed attack algorithms in terms of the approximation error of the local gradient and the distances to the optimal value and optimal point of the local objective.

The rest of this paper is organized as follows. Sec. II defines the problem and gives preliminaries. Sec. III and IV present the design and the analysis of the proposed algorithms. Simulations are provided in Sec. V. Finally, Section VI concludes this paper and discusses future directions.

II. Problem Formulation And Preliminaries

Consider a network described by an undirected, connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote the set of agents and the set of edges, respectively, and $(i, j) \in \mathcal{E}$ if and only if agent i can communicate with agent j. For each agent $i \in \mathcal{V}$, we define $\mathcal{N}_i$ as the set of its neighbors and use $d_i = |\mathcal{N}_i|$ to denote its degree. Let N be the number of agents in $\mathcal{G}$.

A. Problem Formulation

All the agents in the network aim to collaboratively solve the following problem

$$\min_{x\in\mathbb{R}^n} f(x) \triangleq \sum_{i=1}^{N} f_i(x), \tag{1}$$

via local computations and communication. In (1), $f : \mathbb{R}^n \to \mathbb{R}$ is the global objective function, and $f_i(x)$ is the local objective function of agent i and is only known by itself.

We assume that the majority of the agents are trustworthy and faithfully participate in cooperative optimization. However, there are some adversarial agents in $\mathcal{N}_c$ that may deviate from the predefined update rules and aim to infer the sensitive information of agent c, which is some critical agent in the network. To be exact, the sensitive information that these adversaries desire is

$$\mathcal{I}_c \triangleq \{\nabla f_c(x), x_c^*\},$$

where $\nabla f_c(x)$ is the gradient of $f_c(x)$ and x_c^* is the minimizer of $f_c(x)$. Without loss of generality, we assume that agent m is the only malicious agent. If there exist multiple adversaries, then the following designs can be extended likewise. As one of the neighbors of agent c, agent m usually have some common neighbors with agent c, i.e., $\mathcal{N}_c \cap \mathcal{N}_m \neq \varnothing$. These common neighbors can serve as a source of information for agent m to infer $\mathcal{I}_c$.

Let $\mathcal{A}$ be the set of all the common neighbors of agent c and agent m, i.e., $\mathcal{A} \triangleq \mathcal{N}_c \cap \mathcal{N}_m$. Agent m is able to obtain part of the information that is utilized by agent c from $\mathcal{A}$, since when agent $j \in \mathcal{A}$ sends x_j^t to agent c at iteration t, x_j^t is also sent to agent m at the same time. Some basic assumptions are given as follows.

Assumption 1. *Every $f_i(x)$ is continuously differentiable with L-Lipschitz continuous gradients, i.e., $\forall u, v \in \mathbb{R}^n$,*

$$\|\nabla f_i(u) - \nabla f_i(v)\| \leq L\|u - v\|.$$

Assumption 2. *Every $f_i(x)$ is ζ-strongly convex on $\mathbb{R}^n$.*

Assumptions 1 and 2 are satisfied by typical problems in applications, and they are also commonly made in the literature, e.g., [13]–[15].

B. Distributed Gradient Descent

To highlight the central ideas and designs, we consider inference attack algorithms specifically against a classic gradient-based distributed optimization algorithm, i.e., Distributed Gradient Descent (DGD) [12]. As for the inference attacks against other privacy-preserving and resilient distributed optimization algorithms, the proposed algorithms can be extended likewise.

The key idea of DGD is combining gradient descent [16] and average consensus [17]. It requires that each agent i maintains a local state variable x_i, which is an estimation of the global variable x. At iteration t ($t \in \mathbb{N}$), each agent i sends x_i^t to its neighbors $j \in \mathcal{N}_i$, receives state variables x_j^t from them, and decides x_i^{t+1} based on x_j^t ($j \in \mathcal{N}_i$) and the local gradient according to

$$x_i^{t+1} = \sum_{j\in\mathcal{N}_i} w_{ij} x_j^t - \alpha^t \nabla f_i(x_i^t), \tag{2}$$

where $w_{ij} \in (0, 1)$ is the weight, and α^t is the step size used at iteration t. To ensure the effectiveness of (2), $W \triangleq (w_{ij}) \in \mathbb{R}^{n\times n}$ needs to be a doubly stochastic matrix. A classic way of setting W is using lazy Metropolis weights [1], i.e.,

$$w_{ij} = \begin{cases} (2\max(d_i, d_j))^{-1}, & j \in \mathcal{N}_i, j \neq i, \\ 0, & j \notin \mathcal{N}_i, j \neq i, \\ 1 - \sum_{j\in\mathcal{N}_i} w_{ij}, & j = i. \end{cases} \tag{3}$$

With (3), agents need to exchange both the state variables and the degrees during the iterations.

III. Attack with Full Information on $\mathcal{N}_c$

In this section, we consider the case where the adversarial agent m has access to full information on $\mathcal{N}_c$ and design algorithms to approximately infer $\mathcal{I}_c$.

A. Inference Based on Lagrange Interpolation

Lagrange Interpolation [18] uses a smooth polynomial to approximate a function, assuming that the function values at several discrete points are known. Armed with polynomial interpolation, agent m can easily obtain an approximation of $\nabla f_c(x)$ based on the update protocols of agents in the network. Without altering its own update protocol, agent m seems to be normal in terms of its input and output, making it undistinguishable among agents in the network. The details are given as follows.

• *Algorithm Design*

Suppose all the agents in the network iterate with (2), while agent m also records all the $x_j^t, j \in \mathcal{N}_c$ at every iteration t. To simplify the fomulation and emphasize the idea, in this part we let $x \in \mathbb{R}$. For the general case of $x \in \mathbb{R}^n, n \geq 2$, we can usemultivariate interpolation to derive a high-dimensional extension of the proposed algorithm. We define

$$F_c(x) \triangleq \nabla f_c(x), \quad y_c^t \triangleq F_c(x_c^t).$$

It follows from (2) that

$$y_c^t = \frac{\sum_{j\in\mathcal{N}_c} w_{cj} x_j^t - x_c^{t+1}}{\alpha^t}. \tag{4}$$

During the iterations, agents need to exchange both the state variables and the degrees, which makes w_{cj} public in $\mathcal{N}_c$. Thus, (4) means any neighbor of agent c, including agent m, can infer the gradient of agent c with the knowledge of all the neighbors of agent c. Based on this idea, Lagrange Interpolation Polynomial [18] is used to approximate the derivative of the function $f_c(x)$.

At iteration T, agent m is able to obtain x_c^t at iteration $t = 1, 2, \cdots, T$ and all the y_c^t at iteration $t = 0, 1, \cdots, T-1$ based on (4). We simply use $(T-1)$ points, or $\{(x_c^t, y_c^t) | t \in \mathcal{T}\}, \mathcal{T} = \{1, 2, \cdots, T-1\}$, to construct the approximation of F_c. Define

$$L_c(x) \triangleq \sum_{h\in\mathcal{T}} \frac{w_c(x) y_c^h}{w_c'(x_c^h)(x - x_c^h)}, \tag{5}$$

where

$$w_c(x) \triangleq \prod_{h\in\mathcal{T}} (x - x_c^h). \tag{6}$$

Then $L_c(x)$ is an approximation of $F_c(x)$. We summarize the details of such an attack in Algorithm 1.

• *Performance Analysis*

We provide the error bound of the aforementioned approximation in the following theorem.

Theorem 1. *Let* $a \triangleq \min_{t\in\mathcal{T}} x_c^t$ *and* $b \triangleq \max_{t\in\mathcal{T}} x_c^t$. *Suppose that* f_c *is* $(T+1)$*-times differentiable and* $\max_x |f_c^{(T+1)}(x)| = M_{T+1} < \infty$. *Then*

$$|F_c(x) - L_c(x)| = \frac{|w_c(x)|}{T!} |F_c^{(T)}(\xi(x))| \leq \frac{(b-a)^T}{T!} M_{T+1}. \tag{7}$$

Proof. Based on Theorem 6.2 in [18], we have the following error expression for the Lagrange Interpolation

$$F_c(x) - L_c(x) = \frac{w_c(x)}{T!} F_c^{(T)}(\xi(x)) = \frac{w_c(x)}{T!} f_c^{(T+1)}(\xi(x)).$$

By the definition (6), we have $\forall x \in [a, b]$,

$$|w_c(x)| \leq (b-a)^T.$$

By referring to the conditions of Theorem 1, we have (7) holds. □

Algorithm 1: Interpolation-based Attack

```
Initialize x_i^0;
for agent i ∈ V do
    for t = 0, 1, 2, ··· , T do
        Send x_i^t to all the neighbors j ∈ N_i, gather
          x_j^t from them, and calculate x_i^{t+1} with (2);
        if i = m then Calculate y_c^t with (4);
    end
    if i = m then Construct L_c(x) with (5);
end
Output: L_c(x).
```

Remark 1. *In privacy-preserving distributed optimization, it is common to perturb local states with random noises, thus reducing the risk of privacy disclosure. In this case, the added noises may actually facilitate adversarial agents to obtain informative points of* $\nabla f_c(x)$ *with changing* x_c^t*, which leads to a better approximation.*

Armed with (7), we can draw a conclusion that simply by iterating with DGD as normal and recording some received information, an agent m can get an approximation of the gradient of the local function of its neighbor agent c with the error bound given by (7).

B. Manipulation Based on DGD Iteration

Under the above settings, agent m is able to get an approximation of $\nabla f_c(x)$ using $(T-1)$ points. It is noted that when agent m faithfully updates its local state with (2) , x_c^t converges to the global minimizer x^* as t increases. Therefore, the approximation function we obtain is based on several points that move towards the global minimizer, thus the approximation is rather accurate near to x^*.

However, we usually hope to obtain higher accuracy when x_c is close to x_c^*, so that we can estimate the local minimizer x_c^* and local minimum $\nabla f_c(x_c^*)$ with relatively high precision. To deal with this problem, we design the following algorithm, by utilizing which, we have $x_c^t \to x_c^*$ as t increases. Thus, the $(T-1)$ points we sample can be close to x_c^*.

• *Algorithm Design*

Suppose all the agents (including agent m) iterate with (2) at iteration t, while agent m also calculates $\widehat{x}_m^t$ with

$$\widehat{x}_m^t = -\sum_{j\in\mathcal{N}_c} \frac{w_{cj}}{w_{cm}} x_j^t + \frac{1}{w_{cm}} x_c^t. \tag{8}$$

In other words, agent m maintains two states, i.e. $\widehat{x}_m^t$ and x_m^t. At every iteration t, agent m sends $\widehat{x}_m^t$ that update with (8) to agent c, and sends x_m^t that update with (2) to other neighbors of agent m, where $\widehat{x}_m^t$ is used to manipulate the iteration of agent c, and x_m^t is used to keep the attack covert.

The aim of agent m is to make agent c believe itself is working with DGD as normal, even though the truth is that agent c is deceived to iterate with Gradient Descent (GD) [16] algorithm, thus agent c slips into broadcasting its information about its local function to all its neighbors (including agent m) unknowingly, i.e., $x_i^t \to x_c^*$.

Besides, with x_m^t updating as normal, agent m seems to be a ordinary agent from the point of view of other agents except agent c. In contrast, it seems to be agent c that is abnormal, on account of the fact that in effect agent c is updating with GD.

Remark 2. *Actually, protocol* (8) *is only one way to manipulate x_c^t in other to trick agent c to converge to its own minimizer. In theory, by altering protocol* (8)*, we can lead agent c to any point we want. For example, by leading agent c to some particular point, we can use the Chebyshev Polynomial Approximation [19] to replace Lagrange Interpolation, the strong boundedness of which may weaken the assumption of Theorem 1 and reduce the approximation error.*

• *Performance Analysis*

With the design, we have the following theorem naturally.

Theorem 2. *If $\mathcal{A} = \mathcal{N}_c$ and Assumptions 1 and 2 hold, with* (8)*, we have*

$$\lim_{t\to\infty} x_i^t = x_c^*, \quad \forall i \in \mathcal{V}.$$

Proof. Note that agent c and all the agents in $\mathcal{N}_c \setminus \{m\}$ update their states by (2), which is equivalent to

$$x_i^{t+1} = x_i^t - \alpha^t \nabla f_i(x_i^t) + \sum_{j\in\mathcal{N}_i} w_{ij} x_j^t - x_i^t.$$

Let i be c. For agent c, since agent m sends $\widehat{x}_m^t$ to c instead of x_m^t, thus

$$x_c^{t+1} = x_c^t - \alpha^t \nabla f_c(x_c^t) + \sum_{j\in\mathcal{N}_c\setminus\{m\}} w_{cj} x_j^t + w_{cm}\widehat{x}_m^t - x_c^t, \tag{9}$$

where $\widehat{x}_m^t$ is updated with (8). Therefore, the update (9) is transformed to

$$x_c^{t+1} = x_c^t - \alpha^t \nabla f_c(x_c^t).$$

The above update rule is exactly GD algorithm. Therefore, $\lim_{t\to\infty} x_c^t = x_c^*$. For agent $i \in \mathcal{N}_c \setminus \{m\}$, based on Proposition 4.3 from [20] , we have $\lim_{t\to\infty} x_i^t = x_c^*$. □

Remark 3. *In theory, agent m needs to collect all the x_j^t ($j \in \mathcal{N}_c$) before it can calculate $\widehat{x}_m^t$ and then send $\widehat{x}_m^t$ to agent c. However, in a real scenario, it is quite common that there is volatile time delay in the network due to the diverse communication distance and the unstable transmission speed. Therefore, agent m may not necessarily be the last one to send information to agent c, thus reducing its risk of being detected.*

The core idea of Theorem 2 is to make agent c mistakenly think all its neighbors are identical to itself, so that agent c "selfishly" converge to the minimizer of its own object function. Because x_c^t needs to be sent to all its neighbors at iteration t, agent c guadually tells all its neighbors (including agent m) about its own minimizer. It shows that with agent m knowing all the x_j^t in $\mathcal{N}_c$, agent m can successfully manipulate agent c to update its state variable with GD by calculating $\widehat{x}_m^t$ with (8) and sending $\widehat{x}_m^t$ to agent c.

Theorem 2 reflects the effectiveness of (8). However, the condition of Theorem 2, or $\mathcal{A} = \mathcal{N}_c$, is not always suitable for real application scenarios. Usually $\mathcal{A} \subset \mathcal{N}_c$, so agent m only knows all the $x_j^t, j \in \mathcal{A}$, making it difficult to use (8) for manipulation. Thus, the following section is presented to design an algorithm for that senario.

IV. Attack with Partial Information on $\mathcal{N}_c$

In this section, we follow the main idea in the previous section and present our algorithm with a new design, where an adversarial agent m desires to obtain sensitive local information of the critical agent c, even if agent m does not have access to full information on $\mathcal{N}_c$. Let $\mathcal{B} \triangleq \mathcal{N}_c \setminus (\{m\} \cup \mathcal{A})$ and then agent m has no access to any agent in $\mathcal{B}$.

For any agent set $S \subset V$, $\forall t \in \mathbb{N}$, we define

$$\bar{x}_S^t \triangleq \frac{\sum_{k\in S} x_k^t}{|S|},$$

which denotes the average of local state variables in S at iteration t. The core idea of our algorithm is to estimate all the unknown local variables with the average of known variables, i.e., to replace $x_j^t (j \in \mathcal{B})$ with $\bar{x}_{\mathcal{A}}^t$. Then, similar to the previous section, we adjust the data agent m sends to agent c to trick it into performing inexact gradient descent [21]. The details are given as follows.

A. Algorithm Design

Suppose agent m aims to infer x_c^*, while the others are trustworthy agents that iterate by (2) as normal. Aiming at making x_c^t converge to x_c^*, agent m maintains a variable

$$\widehat{x}_m^t = \frac{1}{w_{cm}}(x_c^t - \sum_{j\in\mathcal{A}} w_{cj} x_j^t - \sum_{j\in\mathcal{B}} w_{cj}\bar{x}_{\mathcal{A}}^t), \tag{10}$$

where $W = (w_{ij}) \in \mathbb{R}^{n\times n}$ is doubly stochastic, and $w_{cj} = 0, \forall j \notin \mathcal{N}_c$. Thus

$$\sum_{j\in\mathcal{B}} w_{cj} = 1 - \sum_{j\in\mathcal{A}} w_{cj} - w_{cm}.$$

Similar to the previous section, we suppose agent m updates x_m^t with (2) and sends it to neighbors except agent c, and maintains the variable $\widehat{x}_m^t$ only for agent c. Thus, agent c updates its state with

$$x_c^{t+1} = x_c^t - \alpha^t \nabla f_c(x_c^t) + \sum_{j\in\mathcal{N}_c\setminus\{m\}} w_{cj} x_j^t + w_{cm}\widehat{x}_m^t - x_c^t.$$

Algorithm 2: Manipulation-Based Attack

Initialize x_i^0;
for *agent* $i \in \mathcal{V}$ **do**
  for $t = 0, 1, 2, \cdots, T$ **do**
    if $i = m$ **then**
      Send x_m^t to agents $j \in \mathcal{N}_m \setminus \{c\}$, receive x_j^t from agents $j \in \mathcal{N}_m$, and calculate x_m^{t+1} with (2);
      if $\mathcal{A} = \mathcal{N}_c$ **then**
        Calculate $\widehat{x}_m^t$ with (8);
      else
        Calculate $\widehat{x}_m^t$ with (10);
      end
      Send $\widehat{x}_m^t$ to agent c. Record x_c^t;
    end
    if $i \neq m$ **then**
      Send x_i^t to agents $j \in \mathcal{N}_i$, receive x_j^t from them, and calculate x_i^{t+1} with (2);
    end
  end
end
Output: $\{x_c^t\}$, where $x_c^t \to x_c^*$.

For agent c, we define

$$\begin{aligned}\epsilon^t &\triangleq \sum_{j\in\mathcal{N}_c\setminus\{m\}} w_{cj}x_j^t + w_{cm}\widehat{x}_m^t - x_c^t \\ &= \sum_{j\in\mathcal{B}} w_{cj}x_j^t + \sum_{j\in\mathcal{A}} w_{cj}x_j^t + w_{cm}\widehat{x}_m^t - x_c^t.\end{aligned} \tag{11}$$

Then, we have

$$x_c^{t+1} = x_c^t - \alpha^t \nabla f_c(x_c^t) + \epsilon^t. \tag{12}$$

Algorithm 2 summarizes the details of such an attack.

B. Convergence Analysis

In the following theorem, we use ϵ^t and α^t to describe the distance between x_c^t and the target x_c^*, the proof of which is inspired by [21]. The difference is that we use a general step size α^t instead of a fixed step size, thus our theorem has a wider application scenario.

Theorem 3. *Suppose that Assumptions 1 and 2 hold. Let $\{x_c^t\}_{t\geq 0}$ be the sequence of states of agent c when Algorithm 2 is implemented. Assume that $\alpha^t < \min\{\frac{3}{4L}, \frac{\sqrt{2}}{\zeta}\}$, and α^t is non-increasing. Define*

$$r_t \triangleq \Big[\frac{2}{2\alpha^t(1-L\alpha^t)(\zeta\alpha^t)^2} + L\Big]\|\epsilon^t\|^2,$$

$$\eta_t \triangleq \frac{2-(\zeta\alpha^t)^2}{2} \in (0,1), \quad \kappa \triangleq f_c(x_c^0) - f_c(x_c^*),$$

$$g_t \triangleq (\eta_0+1)\eta_0^t\kappa + (r_t + r_{t-1})\frac{1-\eta_0^{t-1}}{1-\eta_0} + r_t\eta_0^t.$$

Then, for any $t \geq 0$, we have

$$|f_c(x_c^{t+1}) - f_c(x_c^t)| \leq g_t,$$

$$\|x_c^{t+1} - x_c^t\| \leq \frac{2}{L}g_t + \frac{\|\epsilon^t\|^2}{2L\alpha^t(1-L\alpha^t)},$$

$$\|x_c^t - x_c^*\| \leq \frac{2g_t}{L\zeta\alpha^t} + \frac{\|\epsilon^t\|^2}{2L\zeta(\alpha^t)^2(1-L\alpha^t)} + \frac{\|\epsilon^t\|}{\zeta\alpha^t}.$$

Proof. With (12), we have

$$\nabla f_c(x_c^t) = \frac{1}{\alpha^t}(x_c^t - x_c^{t+1} + \epsilon^t). \tag{13}$$

With Assumption 1, we have

$$f_c(x_c^{t+1}) - f_c(x_c^t) \leq \nabla f_c(x_c^t)^T(x_c^{t+1} - x_c^t) + \frac{L}{2}\|x_c^{t+1} - x_c^t\|^2, \tag{14}$$

$$f_c(x_c^t) - f_c(x_c^*) \leq \frac{L}{2}\|x_c^t - x_c^*\|^2. \tag{15}$$

With Assumption 2, we have

$$\nabla f_c(x_c^t)^T(x_c^t - x_c^*) \geq \zeta\|x_c^t - x_c^*\|^2. \tag{16}$$

Combining (13) with (14), we obtain

$$\begin{aligned}&f_c(x_c^{t+1}) - f_c(x_c^t) \\ &\leq \frac{L}{2}\|x_c^{t+1} - x_c^t\|^2 + \frac{1}{\alpha^t}(x_c^t - x_c^{t+1} + \epsilon^t)^T(x_c^{t+1} - x_c^t) \\ &\leq (\frac{L}{2} - \frac{1}{\alpha^t})\|x_c^{t+1} - x_c^t\|^2 + \frac{1}{\alpha^t}\|\epsilon^t\| \cdot \|x_c^{t+1} - x_c^t\|.\end{aligned}$$

Since $\alpha^t < \frac{3}{4L} < \frac{1}{L}$, we have

$$\|\epsilon^t\| \cdot \|x_c^{t+1} - x_c^t\| < \frac{1}{2}\Big[\frac{\|\epsilon^t\|^2}{2-2L\alpha^t} + \frac{\|x_c^{t+1} - x_c^t\|^2}{(2-2L\alpha^t)^{-1}}\Big],$$

thus (14) is transformed into

$$f_c(x_c^{t+1}) - f_c(x_c^t) \leq -\frac{L}{2}\|x_c^{t+1} - x_c^t\|^2 + \frac{\|\epsilon^t\|^2}{4\alpha^t(1-L\alpha^t)},$$

and it is equivalent to

$$\|x_c^{t+1} - x_c^t\|^2 \leq \frac{2}{L}\Big[f_c(x_c^t) - f_c(x_c^{t+1}) + \frac{\|\epsilon^t\|^2}{4\alpha^t(1-L\alpha^t)}\Big]. \tag{17}$$

Using (13) and (16), we obtain

$$\begin{aligned}\|x_c^t - x_c^*\|^2 &\leq \frac{1}{\zeta}\nabla f_c(x_c^t)^T(x_c^t - x_c^*) \\ &\leq \frac{1}{\zeta\alpha^t}\|x_c^t - x_c^{t+1} + \epsilon^t\| \cdot \|x_c^t - x_c^*\|,\end{aligned}$$

thus

$$\|x_c^t - x_c^*\| \leq \frac{1}{\zeta\alpha^t}\|x_c^t - x_c^{t+1} + \epsilon^t\|. \tag{18}$$

Now using (15), (17), and (18), we have

$$\begin{aligned}&f_c(x_c^t) - f_c(x_c^*) \leq \frac{L}{2}\|x_c^t - x_c^*\|^2 \\ &\leq \frac{L\|x_c^t - x_c^{t+1} + \epsilon^t\|^2}{2(\zeta\alpha^t)^2} \leq \frac{L\|x_c^t - x_c^{t+1}\|^2 + L\|\epsilon^t\|^2}{(\zeta\alpha^t)^2} \\ &\leq \frac{2\Big[f_c(x_c^t) - f_c(x_c^{t+1})\Big]}{(\zeta\alpha^t)^2} + \Big[\frac{2}{2\alpha^t(1-L\alpha^t)(\zeta\alpha^t)^2} + L\Big]\|\epsilon^t\|^2 \\ &\leq \frac{2}{(\zeta\alpha^t)^2}\Big[f_c(x_c^t) - f_c(x_c^*) - \Big(f_c(x_c^{t+1}) - f_c(x_c^*)\Big)\Big] + r_t,\end{aligned}$$

then we obtain

$$f_c(x_c^{t+1}) - f_c(x_c^*) \leq \eta_t[f_c(x_c^t) - f_c(x_c^*)] + r_t. \quad (19)$$

Applying (19) recursively, we have

$$\begin{aligned} &f_c(x_c^{t+1}) - f_c(x_c^*) \\ &\leq \prod_{h=0}^{t} \eta_{t-h}[f_c(x_c^0) - f_c(x_c^*)] + \sum_{h=0}^{t} r_{t-h} \prod_{k=0}^{h-1} \eta_{t-k}, \\ &\leq \eta_0^{t+1}\kappa + \sum_{h=0}^{t} r_{t-h}\eta_0^h \leq \eta_0^{t+1}\kappa + r_t \frac{1-\eta_0^t}{1-\eta^0}, \end{aligned}$$

thus

$$\begin{aligned} &|f_c(x_c^t) - f_c(x_c^{t+1})| \leq f_c(x_c^t) - f_c(x_c^*) + f(x_c^{t+1}) - f_c(x_c^*) \\ &\leq (\eta_0 + 1)\eta_0^t\kappa + (r_t + r_{t-1})\frac{1-\eta_0^{t-1}}{1-\eta_0} + r_t\eta_0^t = g_t. \end{aligned}$$

Now using (17), we have

$$\begin{aligned} \|x_c^{t+1} - x_c^t\|^2 &\leq \frac{2}{L}\Big|f_c(x_c^t) - f_c(x_c^{t+1})\Big| + \frac{\|\epsilon^t\|^2}{2L\alpha^t(1-L\alpha^t)} \\ &\leq \frac{2}{L}g_t + \frac{\|\epsilon^t\|^2}{2L\alpha^t(1-L\alpha^t)}. \end{aligned}$$

With (18), we obtain

$$\begin{aligned} \|x_c^t - x_c^*\| &\leq \frac{1}{\zeta\alpha^t}\|x_c^t - x_c^{t+1}\| + \frac{\|\epsilon^t\|}{\zeta\alpha^t} \\ &\leq \frac{2g_t}{L\zeta\alpha^t} + \frac{\|\epsilon^t\|^2}{2L\zeta(\alpha^t)^2(1-L\alpha^t)} + \frac{\|\epsilon^t\|}{\zeta\alpha^t}. \quad \square \end{aligned}$$

Armed with Theorem 3, agent m can infer the sensitive information x_c^*, with the error bound determined by ϵ^t and α^t. Under the setting of Theorem 3, We define

$$d_i^t \triangleq \max_{j,k\in\mathcal{N}_i} \|x_j^t - x_k^t\|, \quad (20)$$

which reflects the similarity of the neighbors of agent i at iteration t. To use Theorem 3 in application, we specify α^t to be a decaying step size, where

$$\alpha^t = \begin{cases} 1/t, & t \in \mathbb{N}^+, \\ 1, & t \in \mathbb{Z} \setminus \mathbb{N}^+. \end{cases} \quad (21)$$

Assume that $d_i^t \leq \mathcal{O}(t^{-p})$ for some $p > 2$, then we have the following corollary on the convergence rates.

Corollary 1. *(Sublinear Convergence) If the conditions of Theorem 3 hold, α^t is calculated by* (21)*, and for some $p > 2, \forall t \geq 1, d_c^t \leq \mathcal{O}(1/t^p)$, then we have*

$$\begin{aligned} |f_c(x_c^{t+1}) - f_c(x_c^t)| &\leq \mathcal{O}(t^{3-2p}), \\ \|x_c^{t+1} - x_c^t\| \leq \mathcal{O}(t^{3-2p}), \quad \|x_c^t - x_c^*\| &\leq \mathcal{O}(t^{1-p}). \end{aligned}$$

Proof. By plugging (10) into (11), we have

$$\epsilon^t = \sum_{j\in\mathcal{B}} w_{cj}(x_j^t - \bar{x}_{\mathcal{A}}^t).$$

With the fact that $\mathcal{A} \subset \mathcal{N}_c, \mathcal{B} \subset \mathcal{N}_c$, we have $\forall j \in \mathcal{B}, k \in \mathcal{A}$, there holds

$$\|x_j^t - x_k^t\| \leq \mathcal{O}(\frac{1}{t^p}).$$

Hence, by the definition of ϵ^t, we have $\forall t \geq 1$,

$$\begin{aligned} \|\epsilon^t\| &= \Big\|\sum_{j\in\mathcal{B}} w_{cj}(x_j^t - \bar{x}_{\mathcal{A}}^t)\Big\| \leq \sum_{j\in\mathcal{B}} \big\|w_{cj}(x_j^t - \bar{x}_{\mathcal{A}}^t)\big\| \\ &\leq \sum_{j\in\mathcal{B}} w_{cj}\frac{\sum_{k\in\mathcal{A}}\|x_j^t - x_k^t\|}{|\mathcal{A}|} \leq \sum_{j\in\mathcal{B}} w_{cj}\mathcal{O}(\frac{1}{t^p}) \leq \mathcal{O}(\frac{1}{t^p}). \end{aligned}$$

With (21), we have

$$\eta_0 = 1 - \frac{\zeta^2}{2} \in (0,1), \quad \frac{1-\eta_0^{t-1}}{1-\eta_0} \in (0,1), \quad \eta_0^t \in (0,1),$$

thus

$$\mathcal{O}(g_t) \leq \mathcal{O}(\eta_0^{t+1}) + \mathcal{O}(r_t) \leq \mathcal{O}(\eta_0^{t+1}) + \mathcal{O}(t^3\|\epsilon^t\|^2) \leq \mathcal{O}(t^{3-2p}).$$

With Theorem 3, we obtain

$$\begin{aligned} |f_c(x_c^{t+1}) - f_c(x_c^t)| &\leq \mathcal{O}(g_t) \leq \mathcal{O}(t^{3-2p}), \\ \|x_c^{t+1} - x_c^t\| &\leq \mathcal{O}(g_t) + \mathcal{O}(t\|\epsilon^t\|^2) \leq \mathcal{O}(t^{3-2p}), \\ \|x_c^t - x_c^*\| &\leq \mathcal{O}(tg_t) + \mathcal{O}(t^2\|\epsilon^t\|^2) + \mathcal{O}(t\|\epsilon^t\|) \leq \mathcal{O}(t^{1-p}). \end{aligned}$$

$\square$

V. Numerical Evaluations

We use two experiments to illustrate the performance of the proposed attack algorithms. Consider an undirected graph with $N = 30$ agents. Let $m = 1, c = 15$, which means that the first agent is the attacker, while the 15th agent is the critical agent being attacked. In the implementations, agent m is set as a neighbor of agent c, and for each agent $j \in \mathcal{N}_c \setminus \{m\}$, we connect agent j and agent m with a certain probability ρ. In the full-information scenario, we have $\rho = 1$. In the partial-information scenario, we set $\rho = 0.5$. For the remaining agents, we set the connectivity probability as 0.4. We use the following convex function as the local object function of agent $i \in \mathcal{V} \setminus \{c\}$, i.e.,

$$f_i(x) = a_i e^{b_i x} + c_i e^{-d_i x}, \quad x \in \mathbb{R},$$

where $a_i, b_i, d_i \sim \mathcal{N}(1, 0.5), c_i \sim \mathcal{N}(3, 0.5)$ are drawn from the Gaussian distribution. For agent c, we set $a_c = b_c = d_c = 1, c_c = 5$, i.e.,

$$f_c(x) = e^x + 5e^{-x}, \quad x \in \mathbb{R}.$$

We set the local object function of agent c differently to make x_c^* (the local minimizer of agent c) far from x^* (the global minimizer), so that we can illustrate the effectiveness of the manipulation more clearly.

In the first experiment, we randomly sample $T - 1 = 4$ points from all the points that agent m obtained from agent c to approximate the gradient of the local function of agent c. The result is shown in Fig 1. It indicates that the approximation algorithm is rather accurate when the variable is close to the points we sampled. The approximation performance is unsatisfactory at points away from the sampled points, because it is difficult for agent m to estimate $\nabla f_c(x)$ there due to the lack of information.

In the second experiment, we manipulate agent c to converge to its local minimizer. Fig. 2(a) and Fig. 2(b) illustrate

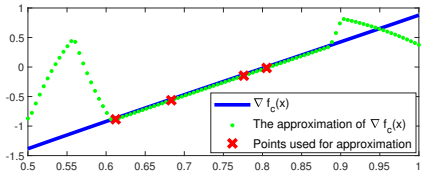
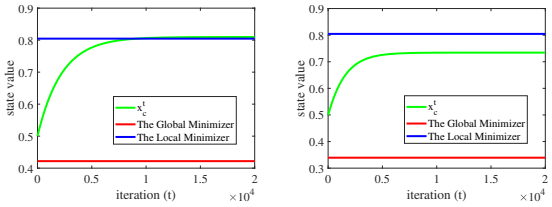


Fig. 1. Simulation results of Algorithm 1.



(a) Full information scenario (b) Partial information scenario

Fig. 2. Simulation results of Algorithm 2 in full-information and partial-information scenarios.

the performance in full-information scenario and partial-information scenario, seperately. They show that when manipulated by agent m, agent c moves towards its local minimizer instead of the global minimizer. Since the attack is more accurate with full information, the performance of the full-information case is better than that of the partial-information case.

VI. Conclusion

We investigated the inference attack based on neighborhood information in distributed optimization, where adversarial agents exploit attack mechanisms to effectively obtain the sensitive information of some critical agent. In the full-information scenario, we presented an algorithm to approximate the gradient of the local objective and trick this critical agent to converge to its own minimizer. In the partial-information scenario, we proposed an attack algorithm based on the average of the collected data and proved that the proposed algorithm can mislead the object agent to its own minimizer with a sublinear convergence rate.

There are many issues worthy of further investigations. In this paper, we assume that the adversaries and the target agent are direct neighbors, i.e., 1-hop neighbors. Thus, we will consider extending our attack algorithms for problems in k-hop neighbors by using information set method [7]. More broadly, research is needed to determine the effect of inference attacks in the presence of other known defenses. Besides, a potential extension is to develop similar attacks for problems with nonconvex local objective functions over time-varying graphs.

References

[1] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[2] J. He, L. Cai, C. Zhao, P. Cheng, and X. Guan, "Privacy-preserving average consensus: Privacy analysis and algorithm design," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 1, pp. 127–138, 2018.

[3] S. Qin, J. He, C. Fang, and J. Lam, "Differential private discrete noise adding mechanism: Conditions, properties and optimization," *arXiv preprint arXiv:2203.10323*, 2022.

[4] J. He, L. Cai, and X. Guan, "Preserving data-privacy with added noises: Optimal estimation and privacy analysis," *IEEE Transactions on Information Theory*, vol. 64, no. 8, pp. 5677–5690, 2018.

[5] M. Ruan, H. Gao, and Y. Wang, "Secure and privacy-preserving consensus," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4035–4049, 2019.

[6] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram, "Resilient asymptotic consensus in robust networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 4, pp. 766–781, 2013.

[7] W. Zheng, Z. He, J. He, and C. Zhao, "Accurate resilient average consensus via detection and compensation," in *Proceedings of the 60th IEEE Conference on Decision and Control*, 2021, pp. 5502–5507.

[8] C. Zhao, J. He, and J. Chen, "Resilient consensus with mobile detectors against malicious attacks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 60–69, 2017.

[9] W. Zheng, Z. He, J. He, C. Zhao, and C. Fang, "Resilient average consensus: A detection and compensation approach," *arXiv preprint arXiv:2202.10814*, 2022.

[10] T. Ding, Q. Xu, S. Zhu, and X. Guan, "A convergence-preserving data integrity attack on distributed optimization using local information," in *Proceedings of the 59th IEEE Conference on Decision and Control*, 2020, pp. 3598–3603.

[11] S. X. Wu, H.-T. Wai, A. Scaglione, A. Nedić, and A. Leshem, "Data injection attack on decentralized optimization," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2018, pp. 3644–3648.

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

[13] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 1245–1260, 2017.

[14] A. Nedic, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.

[15] R. Xin, S. Pu, A. Nedić, and U. A. Khan, "A general framework for decentralized optimization with first-order methods," *Proceedings of the IEEE*, vol. 108, no. 11, pp. 1869–1889, 2020.

[16] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[17] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[18] E. Süli and D. F. Mayers, *An introduction to numerical analysis*. Cambridge University Press, 2003.

[19] L. N. Trefethen, *Approximation Theory and Approximation Practice, Extended Edition*. SIAM, 2019.

[20] S. Sundaram and B. Gharesifard, "Distributed optimization under adversarial nodes," *IEEE Transactions on Automatic Control*, vol. 64, no. 3, pp. 1063–1076, 2018.

[21] A. M.-C. So and Z. Zhou, "Non-asymptotic convergence analysis of inexact gradient methods for machine learning without strong convexity," *Optimization Methods and Software*, vol. 32, no. 4, pp. 963–992, 2017.