Scalable Distributed Optimization of Multi-Dimensional Functions Despite Byzantine Adversaries

Kananart Kuwarananchaoen, *Member, IEEE*, Lei Xin, *Student Member, IEEE*, and Shreyas Sundaram, *Senior Member, IEEE*

***Abstract*—The problem of distributed optimization requires a group of networked agents to compute a parameter that minimizes the average of their local cost functions. While there are a variety of distributed optimization algorithms that can solve this problem, they are typically vulnerable to "Byzantine" agents that do not follow the algorithm. Recent attempts to address this issue focus on single dimensional functions, or assume certain statistical properties of the functions at the agents. In this paper, we provide two resilient, scalable, distributed optimization algorithms for multi-dimensional functions. Our schemes involve two filters, (1) a distance-based filter and (2) a min-max filter, which each remove neighborhood states that are extreme (defined precisely in our algorithms) at each iteration. We show that these algorithms can mitigate the impact of up to F (unknown) Byzantine agents in the neighborhood of each regular agent. In particular, we show that if the network topology satisfies certain conditions, all of the regular agents' states are guaranteed to converge to a bounded region that contains the minimizer of the average of the regular agents' functions.**

***Index Terms*—Byzantine attacks, convex optimization, distributed algorithms, fault tolerance, graph theory, machine learning, multi-agent systems, network security.**

I. Introduction

THE design of distributed algorithms has received significant attention in the past few decades [1], [2]. In particular, for the problem of distributed optimization, a set of agents in a network are required to reach agreement on a parameter that minimizes the average of their local objective functions, using information received from their neighbors [3], [4], [5], [6]. A variety of approaches have been proposed to tackle different challenges of this problem, e.g., distributed optimization under constraints [7], distributed optimization under time-varying graphs [8], and distributed optimization for non-convex non-smooth functions [9]. However, these existing works typically make the assumption that all agents are trustworthy and cooperative (i.e., they follow the prescribed protocol); indeed, such protocols fail if even a single agent behaves in a malicious or incorrect manner [10].

As security becomes a more important consideration in large scale systems, it is crucial to develop algorithms that are resilient to agents that do not follow the prescribed algorithm. A handful of recent papers have considered fault tolerant algorithms for the case where agent misbehavior follows specific patterns [11], [12]. A more general (and serious) form of misbehavior is captured by the *Byzantine* adversary model from computer science, where misbehaving agents can send arbitrary (and conflicting) values to their neighbors at each iteration of the algorithm. Under such Byzantine behavior, it has been shown that it is impossible to guarantee computation of the true optimal point [10], [13]. Thus, researchers have begun formulating distributed optimization algorithms that allow the non-adversarial nodes to converge to a certain region surrounding the true minimizer, regardless of the adversaries' actions [10], [13], [14], [15].

It is worth noting that one major limitation of the above works [10], [13], [14], [15] is that they all make the assumption of scalar-valued objective functions, and the extension of the above ideas to general multi-dimensional convex functions remains largely open. In fact, one major challenge for minimizing multi-dimensional functions is that the region containing the minimizer of the sum of functions is itself difficult to characterize. Specifically, in contrast to the case of scalar functions, where the global minimizer[1] always lies within the smallest interval containing all local minimizers, the region containing the minimizer of the sum of multi-dimensional functions may not necessarily be in the convex hull of the minimizers [16].

There exists a branch of literature focusing on secure distributed machine learning in a client-server architecture [17], [18], [19], where the server appropriately filters the information

Manuscript received 27 July 2023; revised 20 December 2023; accepted 3 March 2024. Date of publication 22 March 2024; date of current version 9 April 2024. This work was supported by the National Science Foundation CAREER under Award 1653648. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Lifeng Lai. *(Corresponding author: Lei Xin.)*

Kananart Kuwarananchaoen was with Purdue University, West Lafayette, IN 47907 USA. He is now with Intel Corporation, Hillsboro, OR 97124 USA (e-mail: kananart.kuwarananchaoen@intel.com).

Lei Xin is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: lxinshenqing@gmail.com).

Shreyas Sundaram is with the Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: sundara2@purdue.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TSIPN.2024.3379844, provided by the authors.

Digital Object Identifier 10.1109/TSIPN.2024.3379844

[1]We will use the terms "global minimizer" and "minimizer of the sum" interchangeably since we only consider convex functions.

received from the clients. However, their extensions to a distributed (peer-to-peer) setting remains unclear. The papers [20], [21], [22] consider a vector version of the resilient machine learning problem in a distributed (peer-to-peer) setting. These papers show that the states of regular nodes will converge to the statistical minimizer with high probability (as the amount of data of each node goes to infinity), but the analysis is restricted to i.i.d training data across the network. However, when each agent has a finite amount of data, these algorithms are still vulnerable to sophisticated attacks as shown in [23]. The work [24] considers a Byzantine distributed optimization problem for multi-dimensional functions, but relies on redundancy among the local functions, and also requires the underlying communication network to be complete. The work presented in [25] proposes a resilient algorithm under statistical characteristic assumptions but lacks guarantees. The recent work [26] studies resilient stochastic optimization problem under non-convex and smooth assumptions on local functions, which differs from our focus. The algorithm proposed in that work achieves convergence to a stationary point up to a constant error but does not ensure asymptotic consensus. Additionally, the recent work [27] offers convergence guarantees to a neighborhood of the optimal solution under deterministic settings, but it pertains to a distinct class of functions – strongly convex and smooth functions.

To the best of our knowledge, our conference paper [28] is the first one that provides a scalable algorithm with convergence guarantees in general networks under very general conditions on the multi-dimensional convex functions held by the agents in the presence of Byzantine faults. Different from existing works, the algorithm in [28] does not rely on any statistical assumptions or redundancy of local functions. Technically, the analysis addresses the challenge of finding a region that contains the global minimizer for multiple-dimensional functions, and shows that regular states are guaranteed to converge to that region under the proposed algorithm. The Distance-MinMax Filtering Dynamics in [28] requires each regular node to compute an auxiliary point using resilient asymptotic consensus techniques on their individual functions' minimizers in advance. After that, there are two filtering steps in the main algorithm that help regular nodes to discard extreme states. The first step is to remove extreme states (based on the distance to the auxiliary point), and the second step is to remove states that have extreme values in any of their components. On the other hand, the algorithm in [28] suffers from the need to compute the auxiliary point prior to running the main algorithm, since the fixed auxiliary point is only achieved by the resilient consensus algorithm asymptotically.

In this paper, we eliminate this drawback. The algorithms and analysis we propose here expand upon the work in [28] in the following significant ways. First, the algorithms in this paper bring the computation of the auxiliary point into the main algorithm, so that the local update of auxiliary point and local filtering strategies are performed simultaneously. This makes the analysis much more involved since we need to take into account the coupled dynamics of the estimated auxiliary point and the optimization variables. Second, the algorithms make better use of local information by including each regular node's own state as a metric. In practice, we observe that this performs better than the approach in [28], since each agent may discard fewer states and hence, there are more non-extreme states that can help the regular agents get close to the true global minimizer. Again, we characterize the convergence region that all regular states are guaranteed to converge to using the proposed algorithm. Third, we present an alternate algorithm in this paper which only makes use of the distance filter (as opposed to both the distance and min-max filter); we show that this algorithm significantly reduces the requirements on the network topology for our convergence guarantees, at the cost of losing guarantees on consensus of the regular nodes' states. Importantly, our work represents the first attempt to provide convergence guarantees in a geometric sense, characterizing a region where all states are ensured to converge to, without relying on any statistical assumptions or redundancy of local functions.

Our paper is organized as follows. Section II introduces various mathematical preliminaries, and states the problem of resilient distributed optimization. We provide our proposed algorithms in Section III. We then state the assumptions and some important results related to properties of the proposed algorithms in Section IV. In Section V, we provide discussion on the results. Finally, we simulate our algorithms to numerically evaluate their performance in Section VI, and conclude in Section VII.

II. Mathematical Notation and Problem Formulation

Let $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{R}$ denote the set of natural numbers (including zero), integers, and real numbers, respectively. We also denote the set of positive integers by $\mathbb{Z}_+$. The cardinality of a set is denoted by $|\cdot|$. The set of subgradients of a convex function f at point x is called the subdifferential of f at x, and is denoted $\partial f(x)$.

A. Linear Algebra

Vectors are taken to be column vectors, unless otherwise noted. We use $x^{(\ell)}$ to represent the ℓ-th component of a vector x. The Euclidean norm on $\mathbb{R}^d$ is denoted by $\|\cdot\|$. We denote by $\langle u, v\rangle$ the Euclidean inner product of u and v, i.e., $\langle u, v\rangle = u^T v$ and by $\angle(u, v)$ the angle between vectors u and v, i.e., $\angle(u, v) = \arccos(\frac{\langle u,v\rangle}{\|u\|\|v\|})$. We use $\mathcal{S}_d^+$ to denote the set of positive definite matrices in $\mathbb{R}^{d\times d}$. The Euclidean ball in d-dimensional space with center at x_0 and radius $r \in \mathbb{R}_{>0}$ is denoted by $\mathcal{B}(x_0, r) := \{x \in \mathbb{R}^d : \|x - x_0\| \le r\}$.

B. Graph Theory

We denote a network by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which consists of the set of nodes $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ and the set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. If $(v_i, v_j) \in \mathcal{E}$, then node v_j can receive information from node v_i. The in-neighbor and out-neighbor sets are denoted by $\mathcal{N}_i^{\text{in}} = \{v_j \in \mathcal{V} : (v_j, v_i) \in \mathcal{E}\}$ and $\mathcal{N}_i^{\text{out}} = \{v_j \in \mathcal{V} : (v_i, v_j) \in \mathcal{E}\}$, respectively. A path from node $v_i \in \mathcal{V}$ to node $v_j \in \mathcal{V}$ is a sequence of nodes $v_{k_1}, v_{k_2}, \ldots, v_{k_l}$ such that $v_{k_1} = v_i$, $v_{k_l} = v_j$ and $(v_{k_r}, v_{k_{r+1}}) \in \mathcal{E}$ for $1 \le r \le l - 1$. Throughout the paper, the terms nodes and agents will be used interchangeably. Given a set of vectors $\{x_1, x_2, \ldots, x_N\}$,

where each $x_i \in \mathbb{R}^d$, we define for all $\mathcal{S} \subseteq \mathcal{V}$,

$$\{x_i\}_{\mathcal{S}} := \{x_i \in \mathbb{R}^d : v_i \in \mathcal{S}\}.$$

Definition 1: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is said to be rooted at node $v_i \in \mathcal{V}$ if for all nodes $v_j \in \mathcal{V} \setminus \{v_i\}$, there is a path from v_i to v_j. A graph is said to be rooted if it is rooted at some node $v_i \in \mathcal{V}$.

We will rely on the following definitions from [29].

Definition 2 (r-reachable set): For a given graph $\mathcal{G}$ and a positive integer $r \in \mathbb{Z}_+$, a subset of nodes $\mathcal{S} \subseteq \mathcal{V}$ is said to be r-reachable if there exists a node $v_i \in \mathcal{S}$ such that $|\mathcal{N}_i^{\text{in}} \setminus \mathcal{S}| \geq r$.

Definition 3 (r-robust graph): For $r \in \mathbb{Z}_+$, a graph $\mathcal{G}$ is said to be r-robust if for all pairs of disjoint nonempty subsets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathcal{V}$, at least one of $\mathcal{S}_1$ or $\mathcal{S}_2$ is r-reachable.

The above definitions capture the idea that sets of nodes should contain individual nodes that have a sufficient number of neighbors outside that set. This will be important for the *local* decisions made by each node in the network under our algorithm, and will allow information from the rest of the network to penetrate into different sets of nodes.

C. Adversarial Behavior

Definition 4: A node $v_i \in \mathcal{V}$ is said to be Byzantine if during each iteration of the prescribed algorithm, it is capable of sending arbitrary (and perhaps conflicting) values to different neighbors. It is also allowed to update its local information arbitrarily at each iteration of any prescribed algorithm.

The set of Byzantine nodes is denoted by $\mathcal{A} \subset \mathcal{V}$. The set of regular nodes is denoted by $\mathcal{R} = \mathcal{V} \setminus \mathcal{A}$.

The identities of the Byzantine agents are unknown to regular agents in advance. Furthermore, we allow the Byzantine agents to know the entire topology of the network, functions equipped by the regular nodes, and the deployed algorithm. In addition, Byzantine agents are allowed to coordinate with other Byzantine agents and access the current and previous information contained by the nodes in the network (e.g. current and previous states of all nodes). Such extreme behavior is typical in the study of the adversarial models [10], [13], [20]. In exchange for allowing such extreme behavior, we will consider a limitation on the number of such adversaries in the neighborhood of each regular node, as follows.

Definition 5 (F-local model): For $F \in \mathbb{Z}_+$, we say that the set of adversaries $\mathcal{A}$ is an F-local set if $|\mathcal{N}_i^{\text{in}} \cap \mathcal{A}| \leq F$, for all $v_i \in \mathcal{R}$.

Thus, the F-local model captures the idea that each regular node has at most F Byzantine in-neighbors.

D. Problem Formulation

Consider a group of N agents $\mathcal{V}$ interconnected over a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each agent $v_i \in \mathcal{V}$ has a local convex cost function $f_i : \mathbb{R}^d \to \mathbb{R}$. The objective is to collaboratively solve the minimization problem

$$\min_{x \in \mathbb{R}^d} \frac{1}{N} \sum_{v_i \in \mathcal{V}} f_i(x), \tag{1}$$

where $x \in \mathbb{R}^d$ is the common decision variable. A common approach to solve such problems is for each agent to maintain a local estimate of the solution to the above problem, which it iteratively updates based on communications with its immediate neighbors. However, since Byzantine nodes are allowed to send arbitrary values to their neighbors at each iteration of any algorithm, it is not possible to solve Problem (1) under such misbehavior (since one is not guaranteed to infer any information about the true functions of the Byzantine agents) [10], [13]. Thus, the optimization problem is recast into the following form:

$$\min_{x \in \mathbb{R}^d} \frac{1}{|\mathcal{R}|} \sum_{v_i \in \mathcal{R}} f_i(x), \tag{2}$$

i.e., we restrict our attention only to the functions held by regular nodes.

Remark 1: In the resilient distributed optimization problem, the agents are required to compute a value that (approximately) minimizes the sum of functions held by each (regular) agent. Compared to the resilient consensus problem, this necessitates more information than simply the initial vectors held by each agent (even if those vectors are initialized to be the local minimizers of the agents' functions). Indeed, the need to combine estimates of the multi-dimensional minimizer from neighbors, while incorporating gradient dynamics, all in a resilient fashion is what makes the resilient distributed optimization problem more difficult than the standard consensus problem.

Remark 2: The additional challenge in solving the above problem lies in the fact that no regular agent is aware of the identities or actions of the Byzantine agents. Furthermore, in the worst-case scenario, it is not feasible to achieve an exact solution to Problem (2), as the Byzantine agents can modify the functions while still adhering to the algorithm, making it impossible to differentiate them [10], [13].

In the next section, we propose two scalable algorithms that allow the regular nodes to approximately solve the above problem, regardless of the identities or actions of the Byzantine agents (as proven later in the paper).

III. Resilient Distributed Optimization Algorithms

A. Proposed Algorithms

The algorithms that we propose are stated as Algorithms 1 and 2. We start with Algorithm 1. At each time-step k, each regular node[2] $v_i \in \mathcal{R}$ maintains and updates a vector $x_i[k] \in \mathbb{R}^d$, which is its estimate of the solution to Problem (2), and a vector $y_i[k] \in \mathbb{R}^d$, which is its estimate of an auxiliary point that provides a general sense of direction for each agent to follow.

Remark 3: The purpose of the estimates $x_i[k]$ is to be an approximation to the minimizer of the sum of the functions. To update this estimate, the agents have to decide which of the estimates provided by their neighbors to retain at each iteration of the algorithm (since up to F of those neighboring estimates may be adversarially chosen by Byzantine agents). To help each regular agent decide which estimates to keep, the auxiliary

[2]Byzantine nodes do not necessarily need to follow the above algorithm, and can update their states however they wish.

Algorithm 1: Simultaneous Distance-MinMax Filtering Dynamics.

Input Network $\mathcal{G}$, functions $\{f_i\}_{i=1}^N$, parameter F

Each $v_i \in \mathcal{R}$ sets $\hat{\boldsymbol{x}}_i^* \leftarrow \texttt{optimize}(f_i)$
Each $v_i \in \mathcal{R}$ sets $\boldsymbol{x}_i[0] \leftarrow \hat{\boldsymbol{x}}_i^*$ and $\boldsymbol{y}_i[0] \leftarrow \hat{\boldsymbol{x}}_i^*$
for $k = 0, 1, 2, 3, \ldots$ **do**
  for $v_i \in \mathcal{R}$ **do** ▷ Implement in parallel
  Step I: Broadcast and Receive
    $\texttt{broadcast}(\mathcal{N}_i^{\text{out}}, \boldsymbol{x}_i[k], \boldsymbol{y}_i[k])$
    $\mathcal{X}_i[k],\ \mathcal{Y}_i[k] \leftarrow \texttt{receive}(\mathcal{N}_i^{\text{in}})$
  Step II: Resilient Consensus Step
    $\mathcal{X}_i^{\text{dist}}[k] \leftarrow \texttt{dist\_filt}(F,\ \boldsymbol{y}_i[k],\ \mathcal{X}_i[k])$
    $\mathcal{X}_i^{\text{mm}}[k] \leftarrow \texttt{x\_minmax\_filt}(F,\ \mathcal{X}_i^{\text{dist}}[k])$
    $\boldsymbol{z}_i[k] \leftarrow \texttt{x\_weighted\_average}(\mathcal{X}_i^{\text{mm}}[k])$
  Step III: Gradient Update
    $\boldsymbol{x}_i[k+1] \leftarrow \texttt{gradient}(f_i, \boldsymbol{z}_i[k])$
  Step IV: Update the Estimated Auxiliary Point
    $\mathcal{Y}_i^{\text{mm}}[k] \leftarrow \texttt{y\_minmax\_filt}(F,\ \mathcal{Y}_i[k])$
    $\boldsymbol{y}_i[k+1] \leftarrow \texttt{y\_weighted\_average}(\mathcal{Y}_i^{\text{mm}}[k])$
  end for
end for

points $\boldsymbol{y}_i[k]$ are used to perform the distance-based filtering step (**Line 7**). In fact, each auxiliary point provides a general sense of direction for the agents' estimates, and thus helps them filter out adversarial estimates that attempt to draw them away from the true minimizer.

We now explain each step used in Algorithm 1 in detail.[3]

- *Line 1:* $\hat{\boldsymbol{x}}_i^* \leftarrow \texttt{optimize}\,(f_i)$
  Each node $v_i \in \mathcal{R}$ uses any appropriate optimization algorithm to get an approximate minimizer $\hat{\boldsymbol{x}}_i^* \in \mathbb{R}^d$ of its local function f_i. We assume that there exists $\epsilon^* \in \mathbb{R}_{\geq 0}$ such that the algorithm achieves $\|\hat{\boldsymbol{x}}_i^* - \boldsymbol{x}_i^*\| \leq \epsilon^*$ for all $v_i \in \mathcal{R}$ where $\boldsymbol{x}_i^* \in \mathbb{R}^d$ is a true minimizer of the function f_i; we assume formally that such a true (but not necessary unique) minimizer exists for each $v_i \in \mathcal{R}$ in the next section.
- *Line 2:* $\boldsymbol{x}_i[0] \leftarrow \hat{\boldsymbol{x}}_i^*$ and $\boldsymbol{y}_i[0] \leftarrow \hat{\boldsymbol{x}}_i^*$
  Each node $v_i \in \mathcal{R}$ initializes its own estimated solution to Problem (2) ($\boldsymbol{x}_i[0] \in \mathbb{R}^d$) and estimated auxiliary point ($\boldsymbol{y}_i[0] \in \mathbb{R}^d$) to be $\hat{\boldsymbol{x}}_i^*$.
- *Line 5:* $\texttt{broadcast}\ (\mathcal{N}_i^{\text{out}}, \boldsymbol{x}_i[k], \boldsymbol{y}_i[k])$
  Node $v_i \in \mathcal{R}$ broadcasts its current state $\boldsymbol{x}_i[k]$ and estimated auxiliary point $\boldsymbol{y}_i[k]$ to its out-neighbors $\mathcal{N}_i^{\text{out}}$.
- *Line 6:* $\mathcal{X}_i[k],\ \mathcal{Y}_i[k] \leftarrow \texttt{receive}(\mathcal{N}_i^{\text{in}})$
  Node $v_i \in \mathcal{R}$ receives the current states $\boldsymbol{x}_j[k]$ and $\boldsymbol{y}_j[k]$ from its in-neighbors $\mathcal{N}_i^{\text{in}}$. So, at time step k, node v_i possesses the sets of states[4]

$$\mathcal{X}_i[k] := \left\{\boldsymbol{x}_j[k] \in \mathbb{R}^d : v_j \in \mathcal{N}_i^{\text{in}} \cup \{v_i\}\right\} \quad \text{and}$$
$$\mathcal{Y}_i[k] := \left\{\boldsymbol{y}_j[k] \in \mathbb{R}^d : v_j \in \mathcal{N}_i^{\text{in}} \cup \{v_i\}\right\}.$$

[3]In the algorithm, $\mathcal{X}_i[k]$, $\mathcal{X}_i^{\text{dist}}[k]$, $\mathcal{X}_i^{\text{mm}}[k]$, $\mathcal{Y}_i[k]$ and $\mathcal{Y}_i^{\text{mm}}[k]$ are multisets.

[4]In case a regular node v_i has a Byzantine neighbor v_j, we abuse notation and take the value $\boldsymbol{x}_j[k]$ to be the value received from node v_j (i.e., it does not have to represent the true state of node v_j).

The sets $\mathcal{X}_i[k]$ and $\mathcal{Y}_i[k]$ have an indirect relationship through the distance-based filter (**Line 7**) as only $\boldsymbol{y}_i[k] \in \mathcal{Y}_i[k]$ is used as the reference to remove states in $\mathcal{X}_i[k]$.

- *Line 7:* $\mathcal{X}_i^{\text{dist}}[k] \leftarrow \texttt{dist\_filt}(F,\ \boldsymbol{y}_i[k],\ \mathcal{X}_i[k])$
  Intuitively, regular node v_i ignores the states that are far away from its own auxiliary state $\boldsymbol{y}_i[k]$ in L^2 sense. Formally, node $v_i \in \mathcal{R}$ computes the distance between each vector in $\mathcal{X}_i[k]$ and its own estimated auxiliary point $\boldsymbol{y}_i[k]$:

$$\mathcal{D}_{ij}[k] := \|\boldsymbol{x}_j[k] - \boldsymbol{y}_i[k]\| \ \text{ for } \ \boldsymbol{x}_j[k] \in \mathcal{X}_i[k]. \tag{3}$$

  Then, node $v_i \in \mathcal{R}$ sorts the values in the set $\{\mathcal{D}_{ij}[k] : v_j \in \mathcal{N}_i^{\text{in}} \cup \{v_i\}\}$ and removes the F largest values that are larger than its own value $\mathcal{D}_{ii}[k]$. If there are fewer than F values higher than its own value, v_i removes all of those values. Ties in values are broken arbitrarily. The corresponding states of the remaining values are stored in $\mathcal{X}_i^{\text{dist}}[k]$. In other words, regular node v_i removes up to F of its neighbors' vectors that are furthest away from the auxiliary point $\boldsymbol{y}_i[k]$.
- *Line 8:* $\mathcal{X}_i^{\text{mm}}[k] \leftarrow \texttt{x\_minmax\_filt}(F,\ \mathcal{X}_i^{\text{dist}}[k])$
  Intuitively, regular node v_i ignores the states that contains extreme values in any of their components in the ordering sense. Formally, for each time-step $k \in \mathbb{N}$ and dimension $\ell \in \{1, 2, \ldots, d\}$, define the set $\mathcal{V}_i^{\text{remove}}(\ell)[k] \subseteq \mathcal{N}_i^{\text{in}}$, where a node v_j is in $\mathcal{V}_i^{\text{remove}}(\ell)[k]$ if and only if
  – $x_j^{(\ell)}[k]$ is within the F-largest values of $\{x_r^{(\ell)}[k] \in \mathbb{R} : \boldsymbol{x}_r[k] \in \mathcal{X}_i^{\text{dist}}[k]\}$ and $x_j^{(\ell)}[k] > x_i^{(\ell)}[k]$, or
  – $x_j^{(\ell)}[k]$ is within the F-smallest values of $\{x_r^{(\ell)}[k] \in \mathbb{R} : \boldsymbol{x}_r[k] \in \mathcal{X}_i^{\text{dist}}[k]\}$ and $x_j^{(\ell)}[k] < x_i^{(\ell)}[k]$.

  Ties in values are broken arbitrarily. Node v_i then removes the state of all nodes in $\bigcup_{\ell \in \{1,2,\ldots,d\}} \mathcal{V}_i^{\text{remove}}(\ell)[k]$ and the remaining states are stored in $\mathcal{X}_i^{\text{mm}}[k]$:

$$\mathcal{X}_i^{\text{mm}}[k] = \Bigg\{ \boldsymbol{x}_j[k] \in \mathbb{R}^d : v_j \in \mathcal{V}_i^{\text{dist}}[k] \setminus \bigcup_{\ell \in \{1,\ldots,d\}} \mathcal{V}_i^{\text{remove}}(\ell)[k] \Bigg\}, \tag{4}$$

  where $\mathcal{V}_i^{\text{dist}}[k] = \{v_j \in \mathcal{R} : \boldsymbol{x}_j[k] \in \mathcal{X}_i^{\text{dist}}[k]\}$.
- *Line 9:* $\boldsymbol{z}_i[k] \leftarrow \texttt{x\_weighted\_average}(\mathcal{X}_i^{\text{mm}}[k])$
  Each node $v_i \in \mathcal{R}$ computes

$$\boldsymbol{z}_i[k] = \sum_{\boldsymbol{x}_j[k] \in \mathcal{X}_i^{\text{mm}}[k]} w_{x,ij}[k]\ \boldsymbol{x}_j[k], \tag{5}$$

  where $w_{x,ij}[k] > 0$ for all $\boldsymbol{x}_j[k] \in \mathcal{X}_i^{\text{mm}}[k]$ and $\sum_{\boldsymbol{x}_j[k] \in \mathcal{X}_i^{\text{mm}}[k]} w_{x,ij}[k] = 1$.
- *Line 10:* $\boldsymbol{x}_i[k+1] \leftarrow \texttt{gradient}\ (f_i, \boldsymbol{z}_i[k])$
  Node $v_i \in \mathcal{R}$ computes the gradient update as follows:

$$\boldsymbol{x}_i[k+1] = \boldsymbol{z}_i[k] - \eta[k]\ \boldsymbol{g}_i[k], \tag{6}$$

  where $\boldsymbol{g}_i[k] \in \partial f_i(\boldsymbol{z}_i[k])$ and $\eta[k]$ is the step-size at time k. The conditions corresponding to the step-size are given in the next section.

Algorithm 2: Simultaneous Distance Filtering Dynamics.

Algorithm 2 is the same as Algorithm 1 except that
- **Line 8** is removed, and
- $\mathcal{X}_i^{\text{mm}}[k]$ in **Line 9** is replaced by $\mathcal{X}_i^{\text{dist}}[k]$.

- *Line 11:* $\mathcal{Y}_i^{\text{mm}}[k] \leftarrow$ `y_minmax_filt`$(F,\ \mathcal{Y}_i[k])$
  For each dimension $\ell \in \{1, 2, \ldots, d\}$, node $v_i \in \mathcal{R}$ removes the F highest and F lowest values of its neighbors' auxiliary points along that dimension. More specifically, for each dimension $\ell \in \{1, 2, \ldots, d\}$, node v_i sorts the values in the set of scalars $\{y_j^{(\ell)}[k] : \boldsymbol{y}_j[k] \in \mathcal{Y}_i[k]\}$ and then removes the F largest and F smallest values that are larger and smaller than its own value, respectively. If there are fewer than F values higher (resp. lower) than its own value, v_i removes all of those values. Ties in values are broken arbitrarily. The remaining values are stored in $\mathcal{Y}_i^{\text{mm}}[k](\ell)$ and the set $\mathcal{Y}_i^{\text{mm}}[k]$ is the collection of $\mathcal{Y}_i^{\text{mm}}[k](\ell)$, i.e., $\mathcal{Y}_i^{\text{mm}}[k] = \{\mathcal{Y}_i^{\text{mm}}[k](\ell) : \ell \in \{1, 2, \ldots, d\}\}$.
- *Line 12:* $\boldsymbol{y}_i[k+1] \leftarrow$ `y_weighted_average`$(\mathcal{Y}_i^{\text{mm}}[k])$
  For each dimension $\ell \in \{1, 2, \ldots, d\}$, each node $v_i \in \mathcal{R}$ computes

$$y_i^{(\ell)}[k+1] = \sum_{y_j^{(\ell)}[k] \in \mathcal{Y}_i^{\text{mm}}[k](\ell)} w_{y,ij}^{(\ell)}[k]\ y_j^{(\ell)}[k], \quad (7)$$

  where $w_{y,ij}^{(\ell)}[k] > 0$ for all $y_j^{(\ell)}[k] \in \mathcal{Y}_i^{\text{mm}}[k](\ell)$ and $\sum_{y_j^{(\ell)}[k] \in \mathcal{Y}_i^{\text{mm}}[k](\ell)} w_{y,ij}^{(\ell)}[k] = 1$.

Note that the filtering process `x_minmax_filt` (**Line 8**) and the filtering process `y_minmax_filt` (**Line 11**) are different. In `x_minmax_filt`, each node removes the **whole** state vector for a neighbor if it contains an extreme value in any component, while in `y_minmax_filt`, each node only removes the extreme components in each vector. In addition, `x_weighted_average` (**Line 9**) and `y_weighted_average_2` (**Line 12**) are also different in that `x_weighted_average` designates agent v_i at time-step k to utilize the same set of weights $\{w_{x,ij} \in \mathbb{R} : \boldsymbol{x}_j[k] \in \mathcal{X}_i^{\text{mm}}[k]\}$ for all components while `y_weighted_average` allows agent v_i at time-step k to use a different set of weights $\{w_{y,ij}^{(\ell)} \in \mathbb{R} : y_j^{(\ell)}[k] \in \mathcal{Y}_i^{\text{mm}}[k](\ell)\}$ for each coordinate ℓ (since the number of remaining values in each component $|\mathcal{Y}_i^{\text{mm}}[k](\ell)|$ is not necessarily the same). These differences will become clear when considering the example provided in the next subsection.

We consider a variant of Algorithm 1 defined as follows.

Although Algorithms 1 and 2 are very similar (differing only in the use of an additional filter in Algorithm 1), our subsequent analysis will reveal the relative costs and benefits of each algorithm. We emphasize that both algorithms involve only simple operations in each iteration, and that the regular agents do not need to know the network topology, or functions possessed by another agents. Furthermore, the regular agents do not need to know the identities of adversaries; they only need to know the upper bound for the number of local adversaries. However, we assume that all regular agents use the same step-size $\eta[k]$ (**Line 10**, (6)).

Remark 4: While the BRIDGE framework, introduced in [21], encompasses several Byzantine-resilient distributed optimization algorithms, including those presented in [10], [13], [14], our proposed algorithms, namely Algorithms 1 and 2, introduce a novel concept of auxiliary states. Specifically, each regular agent v_i in our algorithms maintains an auxiliary state $\boldsymbol{y}_i[k]$, updated using a consensus algorithm, placing them within the broader framework of REDGRAF [27].

While Algorithm 1 from our work shares a similarity with BRIDGE-T [21] by utilizing the coordinate-wise trimmed mean, there are distinctive differences as follows.

- Firstly, our algorithm employs a distance-based filter in addition to the trimmed mean filter, allowing for an asymptotic convergence guarantee under milder assumptions (as provided in Section IV-A). In contrast, the convergence analysis of BRIDGE-T relies on the more restrictive assumption of i.i.d. training data.
- Secondly, the trimmed mean filter in BRIDGE-T eliminates both the smallest and largest F values, whereas our filter discards a subset of these values, similar to the implementation in [10]. This variant in our approach results in faster convergence in practice due to the resulting denser network connectivity after the filtering steps which facilitates quicker information flow [30].
- Lastly, while BRIDGE-T uses a simple average to combine the remaining states, our algorithm employs a weighted average. These weights are chosen to satisfy Assumption 5, ensuring that the weights are lower bounded by a positive constant if the corresponding agents remain after the trimmed mean filter. This provides a more versatile and general scheme.

B. Example of Algorithm 1

Before we prove the convergence properties of the algorithms, we first demonstrate Algorithm 1, which is more complicated due to the min-max filtering step (**Line 8**), step by step using an example.

Suppose there are 8 agents forming the complete graph (for the purpose of illustration). Let node v_i have the local objective function $f_i : \mathbb{R}^2 \to \mathbb{R}$ defined as $f_i(\boldsymbol{x}) = (x^{(1)} + i)^2 + (x^{(2)} - i)^2$ for all $i \in \{1, 2, \ldots, 8\}$. Let the set of adversarial nodes be $\mathcal{A} = \{v_4, v_8\}$ and thus, we have $\mathcal{R} = \{v_1, v_2, v_3, v_5, v_6, v_7\}$. Note that only the regular nodes execute the algorithm (and they do not know which agents are adversarial). Let $F = 2$ and at some time-step $\hat{k} \in \mathbb{N}$, each regular node has the following state and the estimated auxiliary point:[5]

$$\boldsymbol{x}_1[\hat{k}] = \begin{bmatrix} 4 & 2 \end{bmatrix}^T, \quad \boldsymbol{y}_1[\hat{k}] = \begin{bmatrix} 0 & 0 \end{bmatrix}^T,$$

$$\boldsymbol{x}_2[\hat{k}] = \begin{bmatrix} 4 & 1 \end{bmatrix}^T, \quad \boldsymbol{y}_2[\hat{k}] = \begin{bmatrix} -1 & -2 \end{bmatrix}^T,$$

[5]The number of agents in this demonstration is not enough to satisfy the robustness condition (Assumption 4) presented in the next section. However, for our purpose here, it is enough to consider a small number of agents to gain an understanding for each step of the algorithm.

$$\boldsymbol{x}_3[\hat{k}] = \begin{bmatrix}3 & 3\end{bmatrix}^T, \quad \boldsymbol{y}_3[\hat{k}] = \begin{bmatrix}-2 & 1\end{bmatrix}^T,$$
$$\boldsymbol{x}_5[\hat{k}] = \begin{bmatrix}2 & 1\end{bmatrix}^T, \quad \boldsymbol{y}_5[\hat{k}] = \begin{bmatrix}0 & 2\end{bmatrix}^T,$$
$$\boldsymbol{x}_6[\hat{k}] = \begin{bmatrix}1 & 4\end{bmatrix}^T, \quad \boldsymbol{y}_6[\hat{k}] = \begin{bmatrix}1 & 3\end{bmatrix}^T,$$
$$\boldsymbol{x}_7[\hat{k}] = \begin{bmatrix}0 & 0\end{bmatrix}^T, \quad \boldsymbol{y}_7[\hat{k}] = \begin{bmatrix}1 & 3\end{bmatrix}^T.$$

Let $\boldsymbol{x}_{a\to b}[k]$ (resp. $\boldsymbol{y}_{a\to b}[k]$) be the state (resp. estimated auxiliary point) that is sent from the adversarial node $v_a \in \mathcal{A}$ to the regular node $v_b \in \mathcal{R}$ at time-step k. Suppose that in time-step $\hat{k}$, each adversarial agent sends the same states and the same estimated auxiliary points to its neighbors (although this is not necessary) as follows:

$$\boldsymbol{x}_{4\to i}[\hat{k}] = \begin{bmatrix}3 & 2\end{bmatrix}^T, \quad \boldsymbol{y}_{4\to i}[\hat{k}] = \begin{bmatrix}-1 & 1\end{bmatrix}^T,$$
$$\boldsymbol{x}_{8\to i}[\hat{k}] = \begin{bmatrix}0 & 5\end{bmatrix}^T, \quad \boldsymbol{y}_{8\to i}[\hat{k}] = \begin{bmatrix}2 & 2\end{bmatrix}^T$$

for all $i \in \{1,2,3,5,6,7\}$. We will demonstrate the calculation of $\boldsymbol{x}_1[\hat{k}+1]$ and $\boldsymbol{y}_1[\hat{k}+1]$, computed by regular node v_1.

Since the network is the complete graph, the set of in-neighbors and out-neighbors of node v_1 is $\mathcal{N}_1^{\text{in}} = \mathcal{N}_1^{\text{out}} = \mathcal{V} \setminus \{v_1\}$ and $\mathcal{X}_i[\hat{k}]$ (resp. $\mathcal{Y}_i[\hat{k}]$) includes all the states (resp. estimated auxiliary points). Then, node v_1 performs the distance filtering step (**Line 7**) as follows. First, it calculates the squared distances $\mathcal{D}_{1j}^2[\hat{k}]$ (since squaring does not alter the order) for all $\boldsymbol{x}_j[\hat{k}] \in \mathcal{X}_i[\hat{k}]$ as in (3). Node v_1 has

$$\mathcal{D}_{11}^2[\hat{k}] = 20,\ \mathcal{D}_{12}^2[\hat{k}] = 17,\ \mathcal{D}_{13}^2[\hat{k}] = 18,\ \mathcal{D}_{14}^2[\hat{k}] = 13,$$
$$\mathcal{D}_{15}^2[\hat{k}] = 5,\ \mathcal{D}_{16}^2[\hat{k}] = 17,\ \mathcal{D}_{17}^2[\hat{k}] = 0,\ \mathcal{D}_{18}^2[\hat{k}] = 25.$$

Since $\mathcal{D}_{11}^2[\hat{k}]$ is the second largest, node v_1 discards only node v_8's state (which is the furthest away from v_1's auxiliary point) and $\mathcal{X}_1^{\text{dist}}$ contains all states except $\boldsymbol{x}_8[\hat{k}] = \boldsymbol{x}_{8\to 1}[\hat{k}]$.

Then node v_1 performs the min-max filtering process (**Line 8**) as follows. First, consider the first component of the states in $\mathcal{X}_1^{\text{dist}}$. The states of nodes v_1 and v_2 contain the highest value in the first component (which is 4). Since the tie can be broken arbitrarily, we choose $x_1^{(1)}[\hat{k}]$ to come first followed by $x_2^{(1)}[\hat{k}]$ in the ordering, so none of these values are discarded. On the other hand, the state of node v_7 contains the lowest value in its first component, while node v_6's state contains the second lowest value in that component (since node v_8 has already been discarded by the distance filtering process). Node v_1 thus sets $\mathcal{V}_1^{\text{remove}}(1)[\hat{k}] = \{v_6, v_7\}$. Next, consider the second component in which the states of v_6 and v_3 contain the highest and second highest values, respectively, and the states of v_7 and v_5 contain the lowest and second lowest values, respectively. Thus, node v_1 sets $\mathcal{V}_1^{\text{remove}}(2)[\hat{k}] = \{v_3, v_5, v_6, v_7\}$. Since node v_1 removes the entire state from all the nodes in both $\mathcal{V}_1^{\text{remove}}(1)[\hat{k}]$ and $\mathcal{V}_1^{\text{remove}}(2)[\hat{k}]$, according to (4), we have $\mathcal{X}_1^{\text{mm}}[\hat{k}] = \{\boldsymbol{x}_1[\hat{k}], \boldsymbol{x}_2[\hat{k}], \boldsymbol{x}_4[\hat{k}]\} = \{[4\ \ 2]^T, [4\ \ 1]^T, [3\ \ 2]^T\}$.

Next, node v_1 performs the weighted average step (**Line 9**) as follows, Suppose node v_1 assigns the weights $w_{x,11}[\hat{k}] = 0.5$, $w_{x,12}[\hat{k}] = 0.25$ and $w_{x,14}[\hat{k}] = 0.25$. Node v_1 calculates the weighted average according to (5) yielding $z_1^{(1)}[\hat{k}] = 3.75$ and $z_1^{(2)}[\hat{k}] = 1.75$. In the gradient step (**Line 10**), suppose $\eta[\hat{k}] = 0.1$. Node v_1 calculates the gradient of its local function f_1 at $\boldsymbol{z}_1[\hat{k}]$ which yields $\boldsymbol{g}_1[\hat{k}] = [9.5\ \ 1.5]^T$ and then calculates the state $\boldsymbol{x}_1[\hat{k}+1]$ as described in (6) which yields $\boldsymbol{x}_1[\hat{k}+1] = [2.8\ \ 1.6]^T$.

Next, we consider the estimated auxiliary point update of node v_1. In fact, we can perform the update (**Line 11** and **Line 12**) for each component separately. First, consider the first component in which v_8 and v_7 contain the largest and second largest values, respectively, and v_3 and v_2 contain the smallest and second smallest values, respectively. Node v_1 removes these values and thus, $\mathcal{Y}_1^{\text{mm}}[\hat{k}](1) = \{y_1^{(1)}[\hat{k}], y_4^{(1)}[\hat{k}], y_5^{(1)}[\hat{k}], y_6^{(1)}[\hat{k}]\} = \{0, -1, 0, 1\}$. Suppose node v_1 assigns the weights $w_{y,11}^{(1)}[\hat{k}] = w_{y,14}^{(1)}[\hat{k}] = w_{y,15}^{(1)}[\hat{k}] = w_{y,16}^{(1)}[\hat{k}] = 0.25$. Then, the weighted average of the first component according to (7) becomes $y_1^{(1)}[\hat{k}+1] = 0$. Finally, for the second component, v_6 and v_7 contain the largest values, and v_2 and v_1 contain the smallest and second smallest values, respectively. Node v_1 removes the value obtained from v_2, v_6 and v_7 and thus, the set $\mathcal{Y}_1^{\text{mm}}[\hat{k}](2) = \{y_1^{(2)}[\hat{k}], y_3^{(2)}[\hat{k}], y_4^{(2)}[\hat{k}], y_5^{(2)}[\hat{k}], y_8^{(2)}[\hat{k}]\} = \{0, 1, 1, 2, 2\}$. Suppose node v_1 assigns the weights to each value in $\mathcal{Y}_1^{\text{mm}}[\hat{k}](2)$ equally. The weighted average of the second component becomes $y_1^{(2)}[\hat{k}+1] = 1.2$. Thus, we have $\boldsymbol{y}_1[\hat{k}+1] = [0\ \ 1.2]^T$.

IV. Assumptions and Main Results

Having defined the steps in Algorithms 1 and 2, we now turn to proving their resilience and convergence properties.

A. Assumptions

Assumption 1: For all $v_i \in \mathcal{V}$, the functions $f_i(\boldsymbol{x})$ are convex, and the sets $\operatorname{argmin} f_i(\boldsymbol{x})$ are non-empty and bounded.

Since the set $\operatorname{argmin} f_i(\boldsymbol{x})$ is non-empty, let $\boldsymbol{x}_i^*$ be an arbitrary minimizer of the function f_i.

Assumption 2: There exists $L \in \mathbb{R}_{>0}$ such that $\|\tilde{\boldsymbol{g}}_i(\boldsymbol{x})\|_2 \le L$ for all $\boldsymbol{x} \in \mathbb{R}^d$, $v_i \in \mathcal{V}$, and $\tilde{\boldsymbol{g}}_i(\boldsymbol{x}) \in \partial f_i(\boldsymbol{x})$.

The bounded subgradient assumption above is common in the distributed convex optimization literature [31], [32], [33].

Assumption 3: The step-size sequence $\{\eta[k]\}_{k=0}^{\infty} \subset \mathbb{R}_{>0}$ used in **Line 11** of Algorithm 1 is of the form

$$\eta[k] = \frac{c_1}{k + c_2} \quad \text{for some } c_1, c_2 \in \mathbb{R}_{>0}. \tag{8}$$

Note that the step-size in (8) satisfies $\eta[k+1] < \eta[k]$ for all $k \in \mathbb{N}$, and

$$\lim_{k\to\infty} \eta[k] = 0 \quad \text{and} \quad \sum_{k=0}^{\infty} \eta[k] = \infty \tag{9}$$

for any choices of $c_1, c_2 \in \mathbb{R}_{>0}$.

Assumption 4: Given a positive integer $F \in \mathbb{Z}_+$, the Byzantine agents form a F-local set.

Assumption 5: For all $k \in \mathbb{N}$ and $\ell \in \{1, 2, \ldots, d\}$, the weights $w_{x,ij}[k]$ and $w^{(\ell)}_{y,ij}[k]$ (used in **Line 9** and **Line 12** of Algorithm 1) are positive if and only if $x_j[k] \in \mathcal{X}^{\text{mm}}_i[k]$ for Algorithm 1 (and $x_j[k] \in \mathcal{X}^{\text{dist}}_i[k]$ for Algorithm 2) and $y^{(\ell)}_j[k] \in \mathcal{Y}^{\text{mm}}_i[k](\ell)$, respectively. Furthermore, there exists $\omega \in \mathbb{R}_{>0}$ such that for all $k \in \mathbb{N}$ and $\ell \in \{1, 2, \ldots, d\}$, the non-zero weights are lower bounded by ω.

Remark 5: Regarding the prior knowledge of F in Assumption 4, we note that, as with any reliable or secure system, one has to design the system to provide a desired degree of reliability. If one requires the system to provide resilience to a certain number of faulty nodes, one has to design the algorithm (and network) to facilitate that. This is the standard philosophy and methodology in the literature [18], [20], [34]. Note that F does not have to be the exact number of adversarial nodes – it is only an upper bound on the number of adversarial nodes locally.

B. Analysis of Auxiliary Point Update

Since the dynamics of the estimated auxiliary points $\{\boldsymbol{y}_i[k]\}_{\mathcal{R}}$ are independent of the dynamics of the estimated solutions $\{x_i[k]\}_{\mathcal{R}}$, we begin by analyzing the convergence properties of the estimated auxiliary points $\{\boldsymbol{y}_i[k]\}_{\mathcal{R}}$.

In order to establish this result, we need to define the following scalar quantities. For $k \in \mathbb{N}$ and $\ell \in \{1, 2, \ldots, d\}$, let $M^{(\ell)}[k] := \max_{v_i \in \mathcal{R}} y^{(\ell)}_i[k]$, $m^{(\ell)}[k] := \min_{v_i \in \mathcal{R}} y^{(\ell)}_i[k]$, and $D^{(\ell)}[k] := M^{(\ell)}[k] - m^{(\ell)}[k]$. Define the vector $\boldsymbol{D}[k] := [D^{(1)}[k], D^{(2)}[k], \ldots, D^{(d)}[k]]^T$.

The proposition below shows that the estimated auxiliary points $\{\boldsymbol{y}_i[k]\}_{\mathcal{R}}$ converge **exponentially fast** to a single point called $\boldsymbol{y}[\infty]$.

Proposition 1: Suppose Assumption 4 hold, the graph $\mathcal{G}$ is $(2F+1)$-robust, and the weights $w^{(\ell)}_{y,ij}[k]$ satisfy Assumption 5. Suppose the estimated auxiliary points of the regular agents $\{\boldsymbol{y}_i[k]\}_{\mathcal{R}}$ follow the update rule described as **Line 11** and **Line 12** in Algorithm 1. Then, in both Algorithms 1 and 2, there exists $\boldsymbol{y}[\infty] \in \mathbb{R}^d$ with $y^{(\ell)}[\infty] \in [m^{(\ell)}[k], M^{(\ell)}[k]]$ for all $k \in \mathbb{N}$ and $\ell \in \{1, 2, \ldots, d\}$ such that for all $v_i \in \mathcal{R}$, we have

$$\|\boldsymbol{y}_i[k] - \boldsymbol{y}[\infty]\| < \beta e^{-\alpha k},$$

where $\alpha := \frac{1}{|\mathcal{R}|-1} \log \frac{1}{\gamma} > 0$, $\beta := \frac{1}{\gamma}\|\boldsymbol{D}[0]\|$, and $\gamma := 1 - \frac{\omega^{|\mathcal{R}|-1}}{2}$.

The proof of the above proposition follows by noting that the updates for $\{\boldsymbol{y}_i[k]\}_{\mathcal{R}}$ essentially boil down to a set of d scalar consensus updates (one for each dimension of the vector), Thus, one can directly leverage the proof for scalar consensus (with filtering of extreme values) from [10, Proposition 6.3]. Although we extend that proof to provide the explicit convergence rate in Proposition 1, we omit the proof here.

Recall that $\{\hat{x}^*_i\}_{\mathcal{R}}$ is the set containing the approximate minimizers of the regular nodes' local functions. Let x be a matrix in $\mathbb{R}^{d \times |\mathcal{R}|}$, where each column of x is a different vector from $\{\hat{x}^*_i\}_{\mathcal{R}}$. In addition, let $\overline{\boldsymbol{x}}$ and $\underline{\boldsymbol{x}}$ be the vectors in $\mathbb{R}^d$ defined by $\overline{x}_i = \max_{1 \le j \le |\mathcal{R}|}[\boldsymbol{x}]_{ij}$ and $\underline{x}_i = \min_{1 \le j \le |\mathcal{R}|}[\boldsymbol{x}]_{ij}$, respectively. Since we set $\boldsymbol{y}_i[0] = \hat{x}^*_i$ for all $v_i \in \mathcal{R}$ according to **Line 2** in Algorithm 1, we can write

$$\beta = \frac{1}{\gamma}\|\boldsymbol{D}[0]\| = \frac{1}{\gamma}\|\overline{\boldsymbol{x}} - \underline{\boldsymbol{x}}\|.$$

C. Convergence to Consensus of States

Having established convergence of the auxiliary points to a common value (for the regular nodes), we now consider the state update and show that the states of all regular nodes $\{x_i[k]\}_{\mathcal{R}}$ asymptotically reach consensus under Algorithm 1. Before stating the main theorem, we provide a result from [10, Lemma 2.3] which is important for proving the main theorem.

Lemma 1: Suppose the graph $\mathcal{G}$ satisfies Assumption 4 and is $((2\,d+1)F+1)$-robust. Let $\mathcal{G}'$ be a graph obtained by removing $(2\,d+1)F$ or fewer incoming edges from each node in $\mathcal{G}$. Then $\mathcal{G}'$ is rooted.

This means that if we have enough redundancy in the network (in this case, captured by the $((2\,d+1)F+1)$-robustness condition), information from at least one node can still flow to the other nodes in the network even after each regular node discards up to F neighboring states in the distance filtering step (**Line 7**) and up to $2dF$ neighboring states in the min-max filtering step (**Line 8**). This transmissibility of information is a crucial condition for reaching consensus among regular nodes.

Theorem 1 (Consensus): Suppose Assumptions 2–5 hold, and the graph $\mathcal{G}$ is $((2\,d+1)F+1)$-robust. If the regular agents follow Algorithm 1 then for all $v_i, v_j \in \mathcal{R}$, it holds that

$$\lim_{k \to \infty} \|x_i[k] - x_j[k]\| = 0.$$

Proof: It is sufficient to show that all regular nodes $v_i \in \mathcal{R}$ reach consensus on each component of their vectors $\boldsymbol{x}_i[k]$ as $k \to \infty$. For all $\ell \in \{1, 2, \ldots, d\}$ and for all $v_i \in \mathcal{R}$, from (5) and (6), the ℓ-th component of the vector $\boldsymbol{x}_i[k]$ evolves as

$$x^{(\ell)}_i[k+1] = \sum_{\boldsymbol{x}_j[k] \in \mathcal{X}^{\text{mm}}_i[k]} w_{x,ij}[k]\; x^{(\ell)}_j[k] - \eta[k]\; g^{(\ell)}_i[k].$$

From [10, Proposition 5.1], the above equation can be rewritten as

$$x^{(\ell)}_i[k+1] = \sum_{v_j \in (\mathcal{N}^{\text{in}}_i \cap \mathcal{R}) \cup \{v_i\}} \bar{w}^{(\ell)}_{x,ij}[k]\; x^{(\ell)}_j[k] - \eta[k]\; g^{(\ell)}_i[k], \tag{10}$$

where $\bar{w}^{(\ell)}_{x,ii}[k] + \sum_{v_j \in \mathcal{N}^{\text{in}}_i \cap \mathcal{R}} \bar{w}^{(\ell)}_{x,ij}[k] = 1$, and $\bar{w}^{(\ell)}_{x,ii}[k] > \omega$ and at least $|\mathcal{N}^{\text{in}}_i| - 2F$ of the other weights are lower bounded by $\frac{\omega}{2}$.

Consider the set $\mathcal{X}^{\text{mm}}_i[k]$ which is obtained by removing at most $F + 2dF$ states received from v_i's neighbors (up to F states removed by the distance filtering process in **line 7**, and up to $2F$ additional states removed by the min-max filtering process on each of the d components in **line 8**). Since the graph is $((2d+1)F+1)$-robust and the Byzantine agents form an F-local set by Assumption 4, from Lemma 1, the subgraph consisting of regular nodes will be rooted. Using the fact that the term $\eta[k]\; g^{(\ell)}_i[k]$

asymptotically goes to zero (by Assumption 2 and (9)) and (10), we can proceed as in the proof of [10, Theorem 6.1] to show that

$$\lim_{k\to\infty} |x_i^{(\ell)}[k] - x_j^{(\ell)}[k]| = 0,$$

for all $v_i, v_j \in \mathcal{R}$, which completes the proof. □

Theorem 1 established consensus of the states of the regular agents, leveraging (and extending) similar analysis for scalar functions from [10], only for Algorithm 1. However, this does not hold for Algorithm 2 since there might exist a regular agent $v_i \in \mathcal{R}$, time-step $k \in \mathbb{N}$ and dimension $\ell \in \{1, 2, \ldots, d\}$ such that an adversarial state $x_s^{(\ell)}[k] \in \{x_j^{(\ell)}[k] \in \mathbb{R} : \boldsymbol{x}_j[k] \in \mathcal{X}_i^{\text{dist}}[k], v_j \in \mathcal{A}\}$ cannot be written as a convex combination of $\{x_j^{(\ell)}[k] \in \mathbb{R} : v_j \in (\mathcal{N}_i^{\text{in}} \cap \mathcal{R}) \cup \{v_i\}\}$, and thus we cannot obtain (10). On the other hand, Proposition 1 established consensus of the auxiliary points, which will be now used to characterize the convergence *region* of both Algorithms 1 and 2.

D. The Region to Which the States Converge

We now analyze the trajectories of the states of the agents under Algorithms 1 and 2. We start with the following result regarding the intermediate state $\boldsymbol{z}_i[k]$ calculated in **Lines 7-9** of Algorithm 1.

Lemma 2: Suppose Assumptions 4 and 5 hold. Furthermore:

- if the regular agents follow Algorithm 1, suppose the graph $\mathcal{G}$ is $((2\,d+1)F+1)$-robust;
- otherwise, if the regular agents follow Algorithm 2, suppose the graph $\mathcal{G}$ is $(2F+1)$-robust.

For all $k \in \mathbb{N}$ and $v_i \in \mathcal{R}$, if there exists $R_i[k] \in \mathbb{R}_{\geq 0}$ such that $\|\boldsymbol{x}_j[k] - \boldsymbol{y}_i[k]\| \leq R_i[k]$ for all $v_j \in (\mathcal{N}_i^{\text{in}} \cap \mathcal{R}) \cup \{v_i\}$ then $\|\boldsymbol{z}_i[k] - \boldsymbol{y}_i[k]\| \leq R_i[k]$.

Proof: Consider the distance filtering step in **Line 7** of Algorithm 1. Recall the definition of $\mathcal{D}_{ij}[k]$ from (3). We will first prove the following claim. For each $k \in \mathbb{N}$ and $v_i \in \mathcal{R}$, there exists $v_r \in (\mathcal{N}_i^{\text{in}} \cap \mathcal{R}) \cup \{v_i\}$ such that for all $\boldsymbol{x}_j[k] \in \mathcal{X}_i^{\text{dist}}[k]$,

$$\|\boldsymbol{x}_j[k] - \boldsymbol{y}_i[k]\| \leq \|\boldsymbol{x}_r[k] - \boldsymbol{y}_i[k]\|,$$

or equivalently, $\mathcal{D}_{ij}[k] \leq \mathcal{D}_{ir}[k]$.

There are two possible cases. First, if the set $\mathcal{X}_i^{\text{dist}}[k]$ contains only regular nodes, we can simply choose $v_r \in (\mathcal{N}_i^{\text{in}} \cap \mathcal{R}) \cup \{v_i\}$ to be the node whose state $\boldsymbol{x}_r[k]$ is furthest away from $\boldsymbol{y}_i[k]$. Next, consider the case where $\mathcal{X}_i^{\text{dist}}[k]$ contains the state of one or more Byzantine nodes. Since node $v_i \in \mathcal{R}$ removes the F states from $\mathcal{N}_i^{\text{in}}$ that are furthest away from $\boldsymbol{y}_i[k]$ (**Line 7**), and there are at most F Byzantine nodes in $\mathcal{N}_i^{\text{in}}$, there is at least one regular state removed by node v_i. Let v_r be one of the regular nodes whose state is removed. We then have $\mathcal{D}_{ir}[k] \geq \mathcal{D}_{ij}[k]$, for all $v_j \in \{v_s \in \mathcal{V} : \boldsymbol{x}_s[k] \in \mathcal{X}_i^{\text{dist}}[k]\}$ which proves the claim.

If Algorithm 1 is implemented, let $\hat{\mathcal{X}}_i[k] = \mathcal{X}_i^{\text{mm}}[k]$ and we have that $\mathcal{X}_i^{\text{mm}}[k] \subseteq \mathcal{X}_i^{\text{dist}}[k]$ due to the min-max filtering step in **Line 8**. If Algorithm 2 is implemented, let $\hat{\mathcal{X}}_i[k] = \mathcal{X}_i^{\text{dist}}[k]$ since **Line 8** is removed. Then, consider the weighted average step in **Line 9**. From (5), we have

$$\boldsymbol{z}_i[k] - \boldsymbol{y}_i[k] = \sum_{\boldsymbol{x}_j[k] \in \hat{\mathcal{X}}_i[k]} w_{x,ij}[k]\ (\boldsymbol{x}_j[k] - \boldsymbol{y}_i[k])\,.$$

Since $\|\boldsymbol{x}_j[k] - \boldsymbol{y}_i[k]\| \leq \|\boldsymbol{x}_r[k] - \boldsymbol{y}_i[k]\|$ for all $\boldsymbol{x}_j[k] \in \hat{\mathcal{X}}_i[k]$ (where v_r is the node identified in the claim at the start of the proof), we obtain

$$\begin{aligned}\|\boldsymbol{z}_i[k] - \boldsymbol{y}_i[k]\| &\leq \sum_{\boldsymbol{x}_j[k] \in \hat{\mathcal{X}}_i[k]} w_{x,ij}[k]\ \|\boldsymbol{x}_j[k] - \boldsymbol{y}_i[k]\| \\ &\leq \sum_{\boldsymbol{x}_j[k] \in \hat{\mathcal{X}}_i[k]} w_{x,ij}[k]\ \|\boldsymbol{x}_r[k] - \boldsymbol{y}_i[k]\|.\end{aligned}$$

Since $v_r \in (\mathcal{N}_i^{\text{in}} \cap \mathcal{R}) \cup \{v_i\}$, by our assumption, we have $\|\boldsymbol{x}_r[k] - \boldsymbol{y}_i[k]\| \leq R_i[k]$. Thus, using the above inequality and Assumption 5, we obtain that $\|\boldsymbol{z}_i[k] - \boldsymbol{y}_i[k]\| \leq R_i[k]$. □

Lemma 2 essentially states that if the set of states $\{\boldsymbol{x}_j[k] : v_j \in (\mathcal{N}_i^{\text{in}} \cap \mathcal{R}) \cup \{v_i\}\}$ is a subset of the local ball $\mathcal{B}(\boldsymbol{y}_i[k], R_i[k])$ then the intermediate state $\boldsymbol{z}_i[k]$ is still in the ball. This is a consequence of using the distance filter (and adding the min-max filter in Algorithm 1 does not destroy this property), and this will play an important role in proving the convergence theorem.

Next, we will establish certain quantities that will be useful for our analysis of the convergence region. For $v_i \in \mathcal{R}$ and $\epsilon > 0$, define

$$\mathcal{C}_i(\epsilon) := \{\boldsymbol{x} \in \mathbb{R}^d : f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}_i^*) + \epsilon\}. \tag{11}$$

For all $v_i \in \mathcal{R}$, since the set $\operatorname{argmin} f_i(\boldsymbol{x})$ is bounded (by Assumption 1), there exists $\delta_i(\epsilon) \in (0, \infty)$ such that

$$\mathcal{C}_i(\epsilon) \subseteq \mathcal{B}(\boldsymbol{x}_i^*, \delta_i(\epsilon)). \tag{12}$$

The following proposition, whose proof is provided in the supplementary material, introduces an angle θ_i which is an upper bound on the angle between the negative of the gradient of f_i at a given point $\boldsymbol{x}$ and the vector $\boldsymbol{x}_i^* - \boldsymbol{x}$.

Proposition 2: If Assumptions 1 and 2 hold then for all $v_i \in \mathcal{R}$ and $\epsilon > 0$, there exists $\theta_i(\epsilon) \in [0, \frac{\pi}{2})$ such that for all $\boldsymbol{x} \notin \mathcal{C}_i(\epsilon)$ and $\tilde{\boldsymbol{g}}_i(\boldsymbol{x}) \in \partial f_i(\boldsymbol{x})$,

$$\angle(-\tilde{\boldsymbol{g}}_i(\boldsymbol{x}),\ \boldsymbol{x}_i^* - \boldsymbol{x}) \leq \theta_i(\epsilon). \tag{13}$$

Before stating the main theorem, we define

$$\tilde{R}_i := \|\boldsymbol{x}_i^* - \boldsymbol{y}[\infty]\|. \tag{14}$$

Furthermore, for all $\xi \in \mathbb{R}_{\geq 0}$ and $\epsilon \in \mathbb{R}_{>0}$, we define the *convergence radius*

$$s^*(\xi, \epsilon) := \max_{v_i \in \mathcal{R}} \left\{ \max\{\tilde{R}_i \sec\theta_i(\epsilon), \tilde{R}_i + \delta_i(\epsilon)\} \right\} + \xi. \tag{15}$$

where $\tilde{R}_i$, $\theta_i(\epsilon)$ and $\delta_i(\epsilon)$ are defined in (14), (13) and (12), respectively. Based on the definition above, we refer to $\mathcal{B}(\boldsymbol{y}[\infty], s^*(\xi, \epsilon))$ as the *convergence ball*.

We now come to the main result of this paper, showing that the states of all the regular nodes will converge to a ball of radius $\inf_{\epsilon>0} s^*(0, \epsilon)$ around the auxiliary point $\boldsymbol{y}[\infty]$ under Algorithms 1 and 2.

Theorem 2 (Convergence): Suppose Assumptions 1–5 hold. Furthermore:

- if the regular agents follow Algorithm 1, suppose the graph $\mathcal{G}$ is $((2\,d+1)F+1)$-robust;

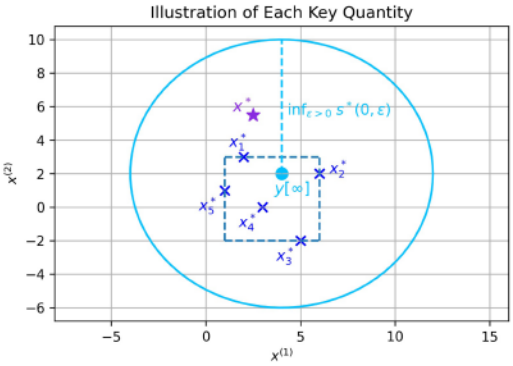


Fig. 1. Local minimizers x_i^* and the global minimizer x^* are shown in the plot. The estimated auxiliary point $y[\infty]$ is in the rectangle formed by the local minimizers (Proposition 1) whereas the global minimizer x^* is not necessarily in the rectangle [16]. However, the ball centered at $y[\infty]$ with radius $\inf_{\epsilon>0} s^*(0,\epsilon)$ contains both the supremum limit of the state vectors $x_i[k]$ and the global minimizer x^* (Theorems 2 and 3).

- otherwise, if the regular agents follow Algorithm 2, suppose the graph $\mathcal{G}$ is $(2F+1)$-robust.

Then regardless of the actions of any F-local set of Byzantine adversaries, for all $v_i \in \mathcal{R}$, we have

$$\limsup_k \|x_i[k] - y[\infty]\| \leq \inf_{\epsilon>0} s^*(0,\epsilon).$$

The proof of the theorem requires several technical lemmas and propositions, and thus, we provide a proof sketch in Section IV-E and a formal proof in the supplementary material.

The following theorem, whose proof is provided in the supplementary material, provides possible locations of the true minimizer x^*, which is in fact inside the convergence region, even in the presence of adversarial agents.

Theorem 3: Let x^* be a solution of Problem (2). If Assumptions 1 and 2 hold, then $x^* \in \mathcal{B}(y[\infty],\ \inf_{\epsilon>0} s^*(0,\epsilon))$.

Theorems 2 and 3 show that both Algorithms 1 and 2 cause all regular nodes to converge to a region that also contains the true solution, regardless of the actions of any F-local set of Byzantine adversaries. The size of this region scales with the quantity $\inf_{\epsilon>0} s^*(0,\epsilon)$. Loosely speaking, this quantity becomes smaller as the minimizers of the local functions of the regular agents get closer together. More specifically, consider a fixed $\epsilon \in \mathbb{R}_{>0}$. If the functions $f_i(x)$ are translated so that the minimizers x_i^* get closer together (i.e., $\tilde{R}_i$ is smaller while $\theta_i(\epsilon)$ and $\delta_i(\epsilon)$ are fixed), then $s^*(0,\epsilon)$ also decreases. Consequently, the state $x_i[k]$ is guaranteed to become closer to the true minimizer x^* as k goes to infinity. Fig. 1 illustrates the key quantities outlined in the main theorems. A detailed discussion of the convergence region is further provided in Section V-D.

We would like to highlight the scalability of our algorithms in terms of both computational complexity and graph robustness requirements, specifically in relation to the number of dimensions d. Algorithms 1 and 2 exhibit computational complexities of $\tilde{\mathcal{O}}(d^2)$ and $\tilde{\mathcal{O}}(d)$ operations per agent per iteration, respectively. A detailed calculation is provided in Section V-C. Furthermore, they impose robustness requirements of $\mathcal{O}(d)$ and $\mathcal{O}(1)$ to achieve the convergence result, as demonstrated in Theorem 2. While Algorithm 2 is more scalable, it lacks a consensus guarantee, unlike Algorithm 1 (refer to Theorem 1). Further insights and discussions on this topic are presented in Section V-B and Remark 6.

E. Proof Sketch of the Convergence Theorem

We work towards the proof of Theorem 2 in several steps, which we provide an overview below. The proofs of the intermediate results presented in this section are provided in the supplementary material.

For the subsequent analysis, we suppose that the graph $\mathcal{G}$

- is $((2\,d+1)F+1)$-robust for Algorithm 1, and
- is $(2F+1)$-robust for Algorithm 2.

Furthermore, unless stated otherwise, we will fix $\xi \in \mathbb{R}_{>0}$ and $\epsilon \in \mathbb{R}_{>0}$, and hide the dependence of ξ and ϵ in $\delta_i(\epsilon)$ and $s^*(\xi,\epsilon)$ by denoting them as δ_i and s^*, respectively.

1) Gradient Update Step Analysis: First, we consider the update from the intermediate states $\{z_i[k]\}_{\mathcal{R}}$ to the states $\{x_i[k+1]\}_{\mathcal{R}}$ via the gradient step (6) (i.e., **Line 10**). In particular, we provide a relationship between $\|z_i[k] - y[\infty]\|$ and $\|x_i[k+1] - y[\infty]\|$ for three different cases:

- $\|z_i[k] - y[\infty]\| \in [0,\ \max_{v_j\in\mathcal{R}}\{\tilde{R}_j + \delta_j\}]$,
- $\|z_i[k] - y[\infty]\| \in (\max_{v_j\in\mathcal{R}}\{\tilde{R}_j + \delta_j\},\ s^*]$,
- $\|z_i[k] - y[\infty]\| \in (s^*, \infty)$.

The corresponding formal statements are presented as follows. Lemma 3 below essentially says that if k is sufficiently large and $z_i[k] \in \mathcal{B}(y[\infty], \max_{v_i\in\mathcal{R}}\{\tilde{R}_i + \delta_i\})$, then after applying the gradient update (6), the state $x_i[k+1]$ will still be in the convergence ball. To establish the result, let $k_1^* \in \mathbb{N}$ be a time-step such that $\eta[k_1^*] \leq \frac{\xi}{L}$.

Lemma 3: Suppose Assumptions 2–5 hold. For all $v_i \in \mathcal{R}$ and $k \geq k_1^*$, if $z_i[k] \in \mathcal{B}(y[\infty],\ \max_{v_j\in\mathcal{R}}\{\tilde{R}_j + \delta_j\})$ then $x_i[k+1] \in \mathcal{B}(y[\infty], s^*)$.

Lemma 4, based on Proposition 2, analyzes the relationship between $\|z_i[k] - y[\infty]\|$ and $\|x_i[k+1] - y[\infty]\|$ when $\|z_i[k] - y[\infty]\| > \tilde{R}_i + \delta_i$. The result will be used to prove Lemma 5.

For $v_i \in \mathcal{R}$, define $\Delta_i : [\tilde{R}_i, \infty) \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ to be the function

$$\Delta_i(p,l) := 2\,l\left(\sqrt{p^2 - \tilde{R}_i^2}\cos\theta_i - \tilde{R}_i \sin\theta_i\right) - l^2. \quad (16)$$

Lemma 4: Suppose Assumptions 1, 2, 4 and 5 hold. For all $v_i \in \mathcal{R}$ and $k \in \mathbb{N}$, if $\|z_i[k] - y[\infty]\| > \tilde{R}_i + \delta_i$ then

$$\|x_i[k+1] - y[\infty]\|^2 \leq \|z_i[k] - y[\infty]\|^2 - \Delta_i(\|z_i[k] - y[\infty]\|,\ \eta[k]\,\|g_i[k]\|), \quad (17)$$

where $g_i[k] \in \mathbb{R}^d$ is defined in (6).

Similar to Lemmas 3, 5 below states that if k is sufficiently large and $\|z_i[k] - y[\infty]\| \in (\max_{v_i\in\mathcal{R}}\{\tilde{R}_i + \delta_i\},\ s^*]$ then by applying the gradient step (6), we have that the state $x_i[k+1]$ is still in the convergence ball.

To simplify the notations, define

$$a_i^{\pm} := -\tilde{R}_i \sin\theta_i \pm \sqrt{(s^*)^2 - \tilde{R}_i^2\cos^2\theta_i} \quad \text{and}$$

$$b_i := 2\left(\sqrt{(s^*)^2 - \tilde{R}_i^2}\cos\theta_i - \tilde{R}_i \sin\theta_i\right). \tag{18}$$

Let $k_2^* \in \mathbb{N}$ be a time-step such that $\eta[k_2^*] \leq \frac{1}{L}\min_{v_i\in\mathcal{R}}\{\min\{a_i^+,\ b_i\}\}$.

Lemma 5: Suppose Assumptions 1–5 hold. For all $v_i \in \mathcal{R}$ and $k \geq k_2^*$, if $\|\boldsymbol{z}_i[k] - \boldsymbol{y}[\infty]\| \in (\max_{v_j\in\mathcal{R}}\{\tilde{R}_j + \delta_j\},\ s^*]$ then $\|\boldsymbol{x}_i[k+1] - \boldsymbol{y}[\infty]\| \in [0,\ s^*]$.

The following lemma is useful for bounding the term Δ_i appeared in (17) for the case that $\|\boldsymbol{z}_i[k] - \boldsymbol{y}[\infty]\| > s^*$.

Define the set of agents

$$\mathcal{I}_z[k] := \{v_i \in \mathcal{R} : \|\boldsymbol{z}_i[k] - \boldsymbol{y}[\infty]\| > s^*\}, \tag{19}$$

and let $k_3^* \in \mathbb{N}$ be a time-step such that $\eta[k_3^*] \leq \frac{1}{2L}\min_{v_i\in\mathcal{R}} b_i$.

Lemma 6: If Assumptions 1–5 hold then for all $k \geq k_3^*$ and $v_i \in \mathcal{I}_z[k]$,

$$\Delta_i(\|\boldsymbol{z}_i[k] - \boldsymbol{y}[\infty]\|,\ \eta[k]\ \|\boldsymbol{g}_i[k]\|) > \frac{1}{2} b_i \underline{L}_i \eta[k],$$

where Δ_i and $\boldsymbol{g}_i[k]$ are defined in (16) and (6), respectively, and $\underline{L}_i := \frac{\epsilon}{\delta_i(\epsilon)} > 0$.

Note that the quantity $\underline{L}_i$ defined above can be interpreted as a lower bound on a subgradient of the function $f_i(\boldsymbol{x})$ when $\boldsymbol{x} \notin \mathcal{C}_i(\epsilon)$.

Lemmas 3–6 collectively establish the complete relationship governing the update from $\{\boldsymbol{z}_i[k]\}_{\mathcal{R}}$ to $\{\boldsymbol{x}_i[k+1]\}_{\mathcal{R}}$, which will be used to prove Lemma 7.

2) Bounds on States of Regular Agents: Next, we consider the update from the states $\{\boldsymbol{x}_i[k]\}_{\mathcal{R}}$ to the intermediate states $\{\boldsymbol{z}_i[k]\}_{\mathcal{R}}$ via two filtering steps (**Lines 7 and 8**) and the weighted average step (**Line 9**). In particular, utilizing Lemma 2, we derive the following relationship.

Proposition 3: If Assumptions 4 and 5 hold, then for all $k \in \mathbb{N}$ and $v_i \in \mathcal{R}$, it holds that

$$\|\boldsymbol{z}_i[k] - \boldsymbol{y}[\infty]\| \leq \max_{v_j\in\mathcal{R}} \|\boldsymbol{x}_j[k] - \boldsymbol{y}[\infty]\| + 2\|\boldsymbol{y}_i[k] - \boldsymbol{y}[\infty]\|.$$

By combining the above inequality with the relationship between $\|\boldsymbol{z}_i[k] - \boldsymbol{y}[\infty]\|$ and $\|\boldsymbol{x}_i[k+1] - \boldsymbol{y}[\infty]\|$ from Lemmas 3–6, and bounding the second term on the RHS, $\|\boldsymbol{y}_i[k] - \boldsymbol{y}[\infty]\|$, using Proposition 1, we obtain a relationship between $\|\boldsymbol{x}_i[k+1] - \boldsymbol{y}[\infty]\|$ and $\max_{v_j\in\mathcal{R}} \|\boldsymbol{x}_j[k] - \boldsymbol{y}[\infty]\|$. As a result, we can bound the distance $\max_{v_i\in\mathcal{R}} \|\boldsymbol{x}_i[k] - \boldsymbol{y}[\infty]\|$ by a particular bounded sequence defined below.

Define the time-step $k_0 \in \mathbb{N}$ as $k_0 := \max_{\ell\in\{1,2,3\}} k_\ell^*$. Recall the definition of α and β from Proposition 1. Let

$$\phi[k_0] = \max_{v_i\in\mathcal{R}} \|\boldsymbol{x}_i[0] - \boldsymbol{y}[\infty]\| + 2\beta \sum_{k=0}^{k_0-1} e^{-\alpha k} + L \sum_{k=0}^{k_0-1} \eta[k], \tag{20}$$

and define a sequence $\{\phi[k]\}_{k=k_0}^{\infty}$ satisfying the update rule

$$\phi^2[k+1] = \max\left\{(s^*)^2,\ \left(\phi[k] + 2\beta e^{-\alpha k}\right)^2 - \frac{1}{2}\eta[k] \min_{v_i\in\mathcal{R}} b_i \underline{L}_i\right\}. \tag{21}$$

Lemma 7: Suppose Assumptions 1–5 hold. For all $k \geq k_0$, it holds that

$$\max_{v_i\in\mathcal{R}} \|\boldsymbol{x}_i[k] - \boldsymbol{y}[\infty]\| \leq \phi[k].$$

Furthermore, there exists $\bar{\phi} \in \mathbb{R}_{\geq 0}$ such that for all $k \geq k_0$, the sequence $\phi[k]$ can be uniformly bounded as $\phi[k] < \bar{\phi}$.

3) Convergence Analysis: Finally, we will utilize the following lemma to further analyze the sequence $\{\phi[k]\}$ defined in (21).

Lemma 8: Consider a sequence $\{\hat{\eta}[k]\}_{k=0}^{\infty} \subset \mathbb{R}_{\geq 0}$ that satisfies $\sum_{k=0}^{\infty} \hat{\eta}[k] = \infty$. If $\gamma_1 \in \mathbb{R}_{\geq 0}$, $\gamma_2 \in \mathbb{R}_{>0}$ and $\lambda \in (-1, 1)$, then there is no sequence $\{u[k]\}_{k=0}^{\infty} \subset \mathbb{R}_{\geq 0}$ that satisfies the update rule

$$u^2[k+1] = (u[k] + \gamma_1\lambda^k)^2 - \gamma_2\hat{\eta}[k].$$

By employing Lemmas 7 and 8, Proposition 4 demonstrates that any repulsion of the state $\boldsymbol{z}_i[k]$ from the convergence ball $\mathcal{B}(\boldsymbol{y}[\infty], s^*)$ due to inconsistency of the estimates of the auxiliary point (Propositions 1 and 3) is compensated by the gradient term pulling the state $\boldsymbol{x}_i[k]$ to the convergence ball. Consequently, the quantity $\phi[k]$ decreases until it does not exceed s^*. In other words, the sequence analysis results in

$$\max_{v_i\in\mathcal{R}} \|\boldsymbol{x}_i[k] - \boldsymbol{y}[\infty]\| \leq \phi[k] \leq s^* \tag{22}$$

for a sufficiently large time-step k. The crucial finite time convergence result is formally stated as follows.

Proposition 4: Suppose Assumptions 1–5 hold. Then, there exists $K \in \mathbb{N}$ such that for all $v_i \in \mathcal{R}$ and $k \geq K$, we have $\boldsymbol{x}_i[k] \in \mathcal{B}(\boldsymbol{y}[\infty], s^*)$.

Since all the prior analyses valid for all $\xi \in \mathbb{R}_{>0}$ and $\epsilon \in \mathbb{R}_{>0}$, the convergence result in Theorem 2 follows from taking $\inf_{\xi>0,\epsilon>0}$ and $\limsup_k$ to (22).

V. Discussion

A. Fundamental Limitation

One would ideally expect an algorithm to provide convergence to the exact minimizer of the sum of the regular agents' functions when there are no Byzantine agents in the network. However, prior works [10], [14] have established a fundamental limitation, showing that achieving such a guarantee *is not possible* unless the set of local functions possesses a redundancy property, known as $2F$-redundancy [24]. This limitation arises from the strong model of Byzantine attacks considered, where a Byzantine agent can substitute the given local function with a forged function that remains legitimate. Consequently, detecting such suspicious behavior or determining the total number of Byzantine agents $|\mathcal{A}|$ in the network is not possible, as the Byzantine agent can follow the algorithm while influencing the outcome of distributed optimization (as discussed in Remark 2). In other words, in settings where Byzantine agents are *potentially* present (i.e., $F > 0$) and there is no known redundancy among the functions, achieving zero steady state error is *impossible* even when there are no Byzantine agents actually present (i.e., $|\mathcal{A}| = 0$) [10], [14].

Our work, imposing only mild assumptions on the local functions, is constrained by this fundamental limit. Although our approach can recover the distributed subgradient method [8], [31] when selecting the parameter $F = 0$, in the worst case scenario, there is no way to determine the number of Byzantine agents. As discussed in Remark 5, in practice, we need to choose the parameter F in the design phase, i.e., prior to the execution of the algorithm. Thus, in our work, the parameter F serves as the maximal number of Byzantine agents in a set of neighbors that the designed system can tolerate, providing a convergence guarantee, as stated in Theorem 2.

It is crucial to acknowledge that the fundamental limit is well-established for distributed optimization problems. However, the question of the dependence of the smallest size of the convergence region on the parameters characterizing the function class remains an important open problem [27].

B. Redundancy and Guarantees Trade-Off

An appropriate notion of network redundancy is necessary for any Byzantine resilient optimization algorithm [10]; for both Algorithms 1 and 2, this is captured by the corresponding robustness conditions in Theorem 2. In particular, Algorithm 1 requires the graph to be $((2\ d + 1)F + 1)$-robust since it implements two filters (a distance-based filter (**Line 7**) and a min-max filter (**Line 8**)) while Algorithm 2 requires the graph to only be $(2F + 1)$-robust as a result of only using the distance-based filter. Since each of these filtering steps discards a set of state vectors, the robustness condition allows the graph to retain some flow of information. Thus, while Algorithm 1 requires significantly stronger conditions on the network topology (i.e., requiring the robustness parameter to scale linearly with the dimension of the functions), it provides the benefit of guaranteeing consensus. Algorithm 2 only requires the robustness parameter to scale with the number of adversaries in each neighborhood, and thus can be used for optimizing high-dimensional functions with relatively sparse networks, at the cost of losing the guarantee on consensus.

Remark 6: The linear dependence of the redundancy requirement on the number of dimensions d is, in fact, typical for resilient vector consensus (e.g., see [35], [36], [37], [38], [39], [40], [41]); The survey paper [42, Section 5.3] provides a detailed discussion of papers that require this assumption. Despite such a condition/restriction being "standard" in the literature, the linear growth in the number of neighbors with the dimension of the state is undesirable. To address the drawback of requiring high redundancy, we provide Algorithm 2 which is an alternative solution that does not depend on the number of dimensions d; however, in this case, we lose the consensus guarantee unlike Algorithm 1.

C. Time Complexity

Suppose the network is r-robust and the number of in-neighbors $|\mathcal{N}_i^{\text{in}}|$ is linearly proportional to r for all $v_i \in \mathcal{V}$. For the distance-based filter (**Line 7**), each regular agent $v_i \in \mathcal{R}$ computes the L^2-norm between its auxiliary state and in-neighbor states and then finds the F agents that attain the maximum value; this procedure takes $\mathcal{O}(dr)$ operations. On the other hand, for the min-max filter (**Line 8**), each regular agent $v_i \in \mathcal{R}$ is required to sort the in-neighbor states for each dimension which takes $\mathcal{O}(dr \log r)$ operations. For Algorithms 1 and 2, the total computational complexities for filtering process are $\tilde{\mathcal{O}}(d^2)$ and $\tilde{\mathcal{O}}(d)$, respectively. Compared to the resilient vector consensus literature [35], [36], [37], [38], [39], [40], [41], which requires exponential in the number of dimensions d for computational complexity, our algorithms have significantly lower computation costs.

D. Convergence Ball

In terms of the size of the convergence ball, it is crucial to note that the convergence radius defined in (15) remains independent of the Lipschitz constant L, the number of regular agents $|\mathcal{R}|$ (in contrast to the result in [26]), or the maximum number of neighboring Byzantine agents F. Instead, the radius hinges solely on specific characteristics of local functions: the locations of local minimizers (captured by $\tilde{R}_i$), sensitivity (captured by θ_i), and the size of the set of local minimizers (captured by δ_i). However, the quantity $\tilde{R}_i$ can be proportional to $\sqrt{d}$ in the worst case as analyzed in [27]. As we will discuss next, remarkably, the sensitivity θ_i defined in Proposition 2 is intimately linked to the condition number of a function.

For simplicity, we will omit the agent index subscript i in the subsequent analysis and assume $x_i^* = \mathbf{0}$. Consider a quadratic function $f(x) = \frac{1}{2}\|Ax\|^2$, where $A \in \mathbb{R}^{d\times d}$ is a positive definite matrix. Now, we will examine the quantity $\sup_{x\neq 0} \angle(g(x), x)$ from Proposition 2, where $g(x) = \nabla f(x) = A^T Ax$. We aim to demonstrate that $\sec(\sup_{x\neq 0} \angle(g(x), x)) \leq (\|A\| \cdot \|A^{-1}\|)^2 := \kappa$, where $\|\cdot\|$ denotes the induced matrix norm, and κ is the condition number associated with the function f [43]. It is noteworthy that this inequality, with the replacement of $\sup_{x\neq 0} \angle(g(x), x)$ by $\sup_{x\neq x_*} \angle(g(x), x - x_*)$, holds for the more general case of $f(x) = \frac{1}{2}\|A(x - x_*) + b\|^2$ with $x_* \in \mathbb{R}^d$ and $b \in \mathbb{R}^d$.

To show such result, we proceed as follows:

$$\begin{aligned}
\cos\left(\sup_{x\neq 0} \angle(g(x), x)\right) &= \inf_{x\neq 0} (\cos \angle(g(x), x)) \\
&= \inf_{x\neq 0} \left(\frac{\langle g(x), x\rangle}{\|g(x)\| \cdot \|x\|}\right) = \inf_{x\neq 0} \left(\frac{\|Ax\|^2}{\|x\|^2} \cdot \frac{\|x\|}{\|A^T Ax\|}\right) \\
&\geq \left(\inf_{x\neq 0} \frac{\|Ax\|}{\|x\|}\right)^2 \left(\sup_{x\neq 0} \frac{\|A^T Ax\|}{\|x\|}\right)^{-1} \\
&\geq \left(\|A^{-1}\| \cdot \|A\|\right)^{-2}.
\end{aligned}$$

In the last inequality, we utilize the properties that $\inf_{x\neq 0} \frac{\|Ax\|}{\|x\|} = \frac{1}{\|A^{-1}\|}$ due to the invertibility of A and $\sup_{x\neq 0} \frac{\|A^T Ax\|}{\|x\|} = \|A^T A\| \leq \|A\|^2$ due to the sub-multiplicative property of induced matrix norm, and $\|A^T\| = \|A\|$.

To get a sense of the convergence region, we consider univariate functions (i.e., the $d = 1$ case). To facilitate the discussion, we denote $\min_{v_i\in\mathcal{R}} x_i^*$ and $\max_{v_i\in\mathcal{R}} x_i^*$ by $\underline{x}$ and $\overline{x}$, respectively. Suppose that the local minimizer x_i^* is unique for all $v_i \in \mathcal{R}$ so

that the quantity δ_i defined in (12) can be chosen arbitrarily close to zero for all $v_i \in \mathcal{R}$. In this case, we have that for all $v_i \in \mathcal{R}$, θ_i defined in (13) is zero. Therefore, the convergence radius s^* in (15) simplifies to $\max_{v_i \in \mathcal{R}} \tilde{R}_i$ (where $\tilde{R}_i$ defined in (14)). In the best case, we can have $y[\infty] = \frac{1}{2}(\underline{x} + \overline{x})$ which results in the convergence region $[\underline{x}, \overline{x}]$ as derived in [10]. In the worst case, (assuming numerical error ϵ^* in **Line 1** is zero) we can have $y[\infty] = \underline{x}$ or $\overline{x}$ which results in the convergence region $[2\underline{x} - \overline{x}, \overline{x}]$ or $[\underline{x}, 2\overline{x} - \underline{x}]$, respectively. In such worst case, the region is two times bigger than the region derived in [10]. These results are due to our "radius analysis" which is uniform in all directions from $y[\infty]$.

Remark 7: Regarding the convergence rate, given the general convex (possibly non-smooth) nature of the problem, achieving only sublinear convergence is typical in centralized settings [44]. Specifically, the anticipated convergence rate may align with the $\mathcal{O}(\frac{\log k}{\sqrt{k}})$ rate observed in [8] for non-faulty distributed cases. While our current work provides asymptotic analysis due to inherent challenges, our future endeavors aim to explore explicit convergence rates for a broader class of Byzantine-resilient distributed optimization algorithms.

E. Maximum Tolerance

Based on the robustness condition for each algorithm and a formula from [45], given the number of agents N in the complete graph and number of dimensions for the optimization variables d, the upper bound on the number of local Byzantine agents F such that the corresponding guarantees still hold, is as follows:

- $F = \lfloor \frac{N-1}{2(2\,d+1)} \rfloor$ for Algorithm 1, and
- $F = \lfloor \frac{1}{4}(N-1) \rfloor$ for Algorithm 2.

From a practical perspective, the robustness property demonstrates a natural trade-off for the system designer. A network that has a stronger robustness property can tolerate more adversaries, but can also induce more costs.

F. Importance of Main States Computation

If we simply implement a resilient consensus protocol on local minimizers similar to the auxiliary states, $y_i[k]$, computation (in **Lines 11-12**) and remove the main states, $x_i[k]$, computation (in **Lines 7-9**), we would obtain that the states of the regular agents converge to the hyper-rectangle formed by the local minimizers (for resilient component-wise consensus algorithms [46]), or the convex hull of the local minimizers (for resilient vector consensus algorithms [39], [47]). Even though using a resilient consensus protocol seems to be a good method for the single dimension case since the resilient distributed optimization algorithm also pushes the states of the regular agents to such sets [10], [13] (and they are identical in this case), it might not give a desired result for the multi-dimensional case. First, it is possible that the minimizer of the sum lies outside both the hyper-rectangle and convex hull [16], [48] as shown in Fig. 1. Second, using only a resilient consensus protocol, one ignores the gradient information which steers the regular agents' states to the true minimizer. Third, we empirically show in Section VI that implementing a resilient distributed optimization algorithm (especially Algorithm 1) usually gives better results (compared to the quality of the solution provided by directly using the auxiliary point, which was obtained by running a resilient consensus protocol on the local minimizers) in terms of both optimality gap and distance to the global minimizer.

G. Importance of Auxiliary States Computation

Essentially, when the main states of regular agents are significantly far away from their local minimizers x_i^*, these minimizers tend to form a *cluster* from the perspective of a regular agent v_i. In addition, building on prior works [16], [48], [49], we know that the true optimal solution x^* (which is the minimizer of the function sum) cannot be located too far away from this cluster. Thus, the auxiliary states y_i, guaranteed to be inside the cluster (in L_1-sense) as shown in Proposition 1, act as valuable references for providing a directional sense to regular agents v_i in their pursuit of the true minimizer x^*. By our design, the distance filter in Algorithms 1 and 2 assumes the role of a guiding mechanism by eliminating extreme states that pull the overall state away from the cluster.

From a technical standpoint, in the multi-dimensional case, relying solely on resilient consensus for the main states x_i and the update using a subgradient g_i with respect to the local function f_i may not suffice to ensure a convergence guarantee. In the worst case, resilient consensus could lead to a state further away from the cluster, especially considering that the strength of this divergence due to Byzantine agents can be proportional to $\sqrt{d}$, where d is the problem dimension. Even though following the subgradient g_i usually mitigates the divergence, it might not be sufficient for guaranteed convergence in such worst cases. Thus, our introduced distance-based filter using a local auxiliary state plays a crucial role in further reducing the severity of the divergence, allowing us to achieve a convergence guarantee under mild assumptions.

VI. Numerical Experiment

We now provide a numerical experiment to illustrate Algorithms 1 and 2. In the experiment, we generate quadratic functions for the local objective functions. Using these functions, we demonstrate the performance (e.g., optimality gaps, distances to the global minimizer) of our algorithms. We also compare the optimality gaps of the function value obtained using the states $x_i[k]$ and the value obtained using the auxiliary points $y_i[k]$, and plot the trajectories of the states of a subset of regular nodes.

Preliminary Settings

- *Main Parameters:* We set the number of nodes to be $n = 25$ and the dimension of each function to be $d = 2$.
- *Adversary Parameters:* We consider the F-local model, and set $F = 2$ for Algorithm 1 and $F = 5$ for Algorithm 2.

Network Settings

- *Topology Generation:* We construct an 11-robust graph on $n = 25$ nodes following the approach from [29], [45]. This graph can tolerate up to 2 local adversaries for Algorithm 1, and up to 5 local adversaries for Algorithm 2 according to Theorem 2. Note that the same graph is used to perform numerical experiments for both Algorithms 1 and 2.

Adversaries' Strategy

- *Adversarial Nodes:* We construct the set of adversarial nodes $\mathcal{A}$ by randomly choosing nodes in $\mathcal{V}$ so that the set of adversarial nodes form a F-local set. Note that in general, constructing $\mathcal{A}$ depends on the topology of the network. In our experiment, we have $\mathcal{A} = \{v_9, v_{16}\}$ for Algorithm 1 and $\mathcal{A} = \{v_5, v_{11}, v_{12}, v_{17}, v_{22}, v_{24}\}$ for Algorithm 2.
- *Adversarial Values Transmitted:* Here, we use a sophisticated approach rather than simply choosing the transmitted values at random. Suppose v_s is an adversary node and v_i is a regular node which is an out-neighbor of v_s, i.e., $v_s \in \mathcal{N}_i^{\text{in}}$. First, consider the state of nodes in the network at time-step k. The adversarial node v_s uses an oracle to determine the region in the state space for the regular node v_i in which if the adversarial node selects the transmitted value to be outside the region then the value will be discarded by that regular agent v_i. Then, v_s chooses $\boldsymbol{x}_{s\to i}[k]$ (the forged state sent from v_s to v_i at time k) so that it is in the safe region and far from the global minimizer. In this way, the adversaries' values will not be discarded and also try to prevent the regular nodes from getting close to the minimizer. Similarly, for the auxiliary point update, the adversarial node v_s uses an oracle to determine the safe region in the auxiliary point's space for the regular node v_i. Since the safe region is a hyper-rectangle in general, v_s chooses $\boldsymbol{y}_{s\to i}[k]$ (the forged estimated auxiliary point sent from v_s to v_i at time k) to be near a corner (chosen randomly) of the hyper-rectangle.

Objective Functions Settings

- *Local Functions:* For $v_i \in \mathcal{V}$, we set the local objective functions $f_i : \mathbb{R}^d \to \mathbb{R}$ to be

$$f_i(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{Q}_i \boldsymbol{x} + \boldsymbol{b}_i^T \boldsymbol{x},$$

  where $\boldsymbol{Q}_i \in \mathcal{S}_d^+$ and $\boldsymbol{b}_i \in \mathbb{R}^d$ are chosen randomly. Note that the same local functions are used to perform numerical experiments for both Algorithms 1 and 2.
- *Global Objective Function:* According to our objective (2), we then have the global objective function $f : \mathbb{R}^d \to \mathbb{R}$ as follows:

$$f(\boldsymbol{x}) = \frac{1}{|\mathcal{R}|}\left(\frac{1}{2}\boldsymbol{x}^T\left(\sum_{v_i\in\mathcal{R}} \boldsymbol{Q}_i\right)\boldsymbol{x} + \left(\sum_{v_i\in\mathcal{R}} \boldsymbol{b}_i\right)^T \boldsymbol{x}\right),$$

  where the set of regular nodes $\mathcal{R} = \mathcal{V} \setminus \mathcal{A}$.

Algorithm Settings

- *Initialization:* For each regular node $v_i \in \mathcal{R}$, we compute the exact minimizer $\boldsymbol{x}_i^* = -\boldsymbol{Q}_i^T \boldsymbol{b}_i$ and use it as the initial state and auxiliary point of v_i as suggested in **Line 1-2** of Algorithm 1.
- *Weights Selection:* For each time-step $k \in \mathbb{N}$ and regular node $v_i \in \mathcal{R}$, we randomly choose the weights $w_{x,ij}[k], w_{y,ij}^{(\ell)}[k]$ so that they follow the description of **Line 9** and **Line 12**, and Assumption 5.
- *Step-size Selection:* We choose the step-size schedule (in **Line 11** of Algorithm 1) to be $\eta[k] = \frac{1}{k+1}$.

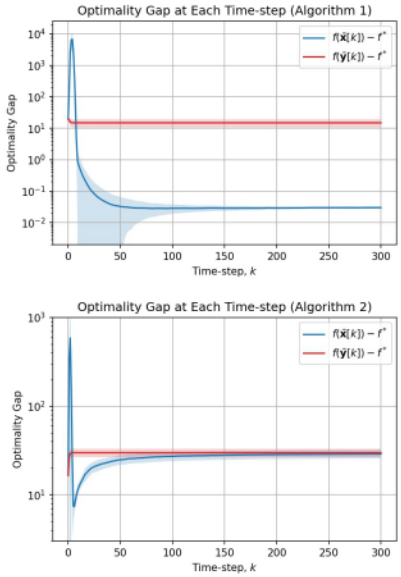


Fig. 2. Plots show the optimality gap evaluated at the average of the regular nodes' states $f(\bar{\boldsymbol{x}}[k]) - f^*$ averaged over 10 runs (blue), and the optimality gap evaluated at the average of the regular nodes' auxiliary points $f(\bar{\boldsymbol{y}}[k]) - f^*$ averaged over 10 runs (red) against the time-step k obtained from (top) Algorithm 1 and (bottom) Algorithm 2. The shaded regions represent +1/-1 standard deviation.

- *Gradient Norm Bound:* We choose the upper bound of the gradient norm to be $L = 10^5$. If the norm exceeds the bound, we scale the gradient down so that its norm is equal to L, i.e.,

$$\boldsymbol{g}_i[k] = \begin{cases} \nabla f_i(\boldsymbol{z}_i[k]) & \text{if } \|\nabla f_i(\boldsymbol{z}_i[k])\| \le L, \\ \frac{L}{\|\nabla f_i(\boldsymbol{z}_i[k])\|} \cdot \nabla f_i(\boldsymbol{z}_i[k]) & \text{otherwise.} \end{cases}$$

Simulation Settings and Results

- *Time Horizon:* We set the time horizon of our simulations to be $K = 300$ (starting from $k = 0$).
- *Experiments Detail:* For both Algorithms 1 and 2, we fix the graph, local functions, and step-size schedule. However, since the set of adversaries are different, the global objective functions, and hence the global minimizers are different. For each algorithm, we run the experiment 10 times setting the same states initialization across the runs. The results from the runs are different due to the randomness in the adversaries' strategy.
- *Performance Metrics:* We examine the performance of our algorithms by considering the optimality gaps (Fig. 2), distances to the global minimizer (Fig. 3), and trajectories of randomly selected regular agents (Fig. 4).
- *Algorithm 1's Results:* The lines corresponding to the optimality gap and distance to the global minimizer evaluated using auxiliary points are almost horizontal since the convergence to consensus is very fast. However, one can see that the optimality gap and distance to the minimizer obtained from the regular states are significantly smaller than that from the auxiliary points due to the use of gradient information (**Line 10**) and extreme states filtering (**Line 8**) in the regular state update. In particular, at $k = 300$, the

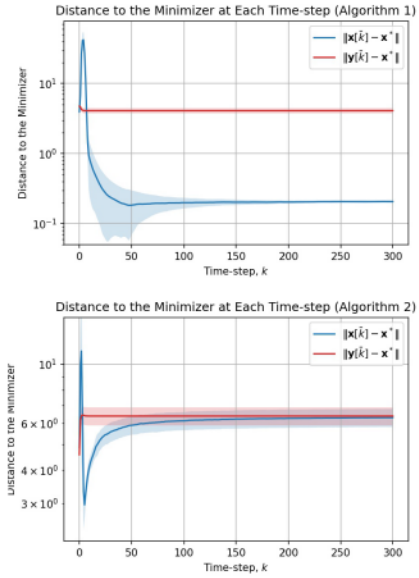


Fig. 3. Plots show the distance between the average of the regular nodes' states and the global minimizer $\|\bar{x}[k] - x^*\|$ averaged over 10 runs (blue), and the distance between the average of the regular nodes' auxiliary points and the global minimizer $\|\bar{y}[k] - x^*\|$ averaged over 10 runs (red) against the time-step k obtained from (top) Algorithm 1 and (bottom) Algorithm 2. The shaded regions represent +1/-1 standard deviation.

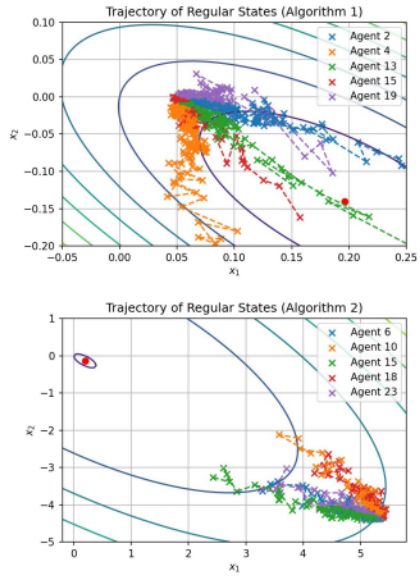


Fig. 4. Plots show the trajectory of the states of a subset of the regular nodes obtained from (top) Algorithm 1 and (bottom) Algorithm 2. Different colors of the trajectory represent different regular agents v_i in the network. In each figure, the contour plot shows the level sets of the global objective function (in this case, a quadratic function) and the red dot represents the global minimizer.

optimality gap and distance to the global minimizer at the regular states' average are only about 0.030 and 0.206, respectively. Moreover, the state trajectories converge together and stay close to the global minimizer even in the presence of sophisticated adversaries. Note that, from our observations, Algorithm 1 yields better results than Algorithm 2 given the same settings.

- *Algorithm 2's Results:* The optimality gaps and distances to the global minimizer evaluated using the states are slightly better than the values obtained using the auxiliary points, and the state trajectories remain reasonably close to the global minimizer showing that the algorithm can tolerate $F = 5$ local adversaries (which is more than Algorithm 1). Interestingly, the state trajectories seem to converge together even though the consensus guarantee is lacking due to the absence of the distance-based filter.

VII. Conclusion and Future Work

In this paper, we considered the distributed optimization problem in the presence of Byzantine agents. We developed two resilient distributed optimization algorithms for multi-dimensional functions. The key improvement over our previous work in [28] is that the algorithms proposed in this paper do not require a fixed auxiliary point to be computed in advance (which will not happen under finite time in general). Our algorithms have low complexity and each regular node only needs local information to execute the steps. Algorithm 1 (with the min-max state filter), which requires more network redundancy, guarantees that the regular states can asymptotically reach consensus and enter a bounded region that contains the global minimizer, irrespective of the actions of Byzantine agents. On the other hand, Algorithm 2 (without the min-max filter) has a more relaxed condition on the network topology and can guarantee asymptotic convergence to the same region, but cannot guarantee consensus. For both algorithms, we explicitly characterized the size of the convergence region, and showed through simulations that Algorithm 1 appears to yield results that are closer to optimal, as compared to Algorithm 2.

As noted earlier, the consensus guarantee for Algorithm 1 requires linear scaling of network robustness with the dimension of the local functions, which can be limiting in practice. This seems to be a common challenge for resilient consensus-based algorithms in systems with multi-dimensional states, e.g., [24], [47], [50]. Finding a relaxed condition on the network topology for high-dimensional resilient distributed optimization problems (with guaranteed consensus) would be a rich area for future research.

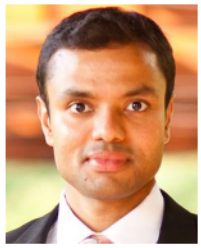
References

[1] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Control*, vol. 31, no. 9, pp. 803–812, Sep. 1986.
[2] L. Xiao and S. Boyd, "Optimal scaling of a gradient method for distributed resource allocation," *J. Optim. Theory Appl.*, vol. 129, no. 3, pp. 469–488, 2006.
[3] J. Wang and N. Elia, "A control perspective for centralized and distributed convex optimization," in *Proc. IEEE 50th Conf. Decis. Control Eur. Control Conf.*, 2011, pp. 3800–3805.
[4] S. Boyd et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.
[5] M. Eisen, A. Mokhtari, and A. Ribeiro, "Decentralized quasi-newton methods," *IEEE Trans. Signal Process.*, vol. 65, no. 10, pp. 2613–2628, May 2017.
[6] R. Xin, C. Xi, and U. A. Khan, "FROST–Fast row-stochastic optimization with uncoordinated step-sizes," *EURASIP J. Adv. Signal Process.*, vol. 2019, no. 1, 2019, Art. no. 1.

[7] M. Zhu and S. Martínez, "On distributed convex optimization under inequality and equality constraints," *IEEE Trans. Autom. Control*, vol. 57, no. 1, pp. 151–164, Jan. 2011.
[8] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Trans. Autom. Control*, vol. 60, no. 3, pp. 601–615, Mar. 2015.
[9] J. Zeng and W. Yin, "On nonconvex decentralized gradient descent," *IEEE Trans. Signal Process.*, vol. 66, no. 11, pp. 2834–2848, Jun. 2018.
[10] S. Sundaram and B. Gharesifard, "Distributed optimization under adversarial nodes," *IEEE Trans. Autom. Control*, vol. 64, no. 3, pp. 1063–1076, Mar. 2019.
[11] N. Ravi, A. Scaglione, and A. Nedić, "A case of distributed optimization in adversarial environment," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2019, pp. 5252–5256.
[12] S. X. Wu, H.-T. Wai, A. Scaglione, A. Nedić, and A. Leshem, "Data injection attack on decentralized optimization," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2018, pp. 3644–3648.
[13] L. Su and N. H. Vaidya, "Byzantine-resilient multiagent optimization," *IEEE Trans. Autom. Control*, vol. 66, no. 5, pp. 2227–2233, May 2021.
[14] L. Su and N. H. Vaidya, "Fault-tolerant multi-agent optimization: Optimal iterative distributed algorithms," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2016, pp. 425–434.
[15] C. Zhao, J. He, and Q.-G. Wang, "Resilient distributed optimization algorithm against adversarial attacks," *IEEE Trans. Autom. Control*, vol. 65, no. 10, pp. 4308–4315, Oct. 2020.
[16] K. Kuwarananchoren and S. Sundaram, "On the location of the minimizer of the sum of two strongly convex functions," in *Proc. IEEE Conf. Decis. Control*, 2018, pp. 1769–1774.
[17] N. Gupta and N. H. Vaidya, "Byzantine fault tolerant distributed linear regression," 2019, *arXiv:1903.08752*.
[18] P. Blanchard et al., "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 119–129.
[19] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Trans. Signal Process.*, vol. 70, pp. 1142–1154, 2022.
[20] Z. Yang and W. U. Bajwa, "ByRDiE: Byzantine-resilient distributed coordinate descent for decentralized learning," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 4, pp. 611–627, Dec. 2019.
[21] C. Fang, Z. Yang, and W. U. Bajwa, "BRIDGE: Byzantine-resilient decentralized gradient descent," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 8, pp. 610–626, 2022.
[22] A. R. Elkordy, S. Prakash, and S. Avestimehr, "Basil: A fast and byzantine-resilient approach for decentralized training," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 9, pp. 2694–2716, Sep. 2022.
[23] S. Guo, T. Zhang, X. Xie, L. Ma, T. Xiang, and Y. Liu, "Towards byzantine-resilient learning in decentralized systems," *IEEE Trans. Circuits Syst. Video Technol.* vol. 32, no 6, pp. 4096–4106, Jun. 2022, doi: 10.1109/TCSVT.2021.3116976.
[24] N. Gupta, T. T. Doan, and N. H. Vaidya, "Byzantine fault-tolerance in decentralized optimization under 2F-redundancy," in *Proc. Amer. Control Conf.*, 2021, pp. 3632–3637.
[25] N. Ravi and A. Scaglione, "Detection and isolation of adversaries in decentralized optimization for non-strongly convex objectives," *IFAC-PapersOnLine*, vol. 52, no. 20, pp. 381–386, 2019.
[26] Z. Wu, T. Chen, and Q. Ling, "Byzantine-resilient decentralized stochastic optimization with robust aggregation rules," *IEEE Trans. Signal Process.*, vol. 71, pp. 3179–3195, 2023.
[27] K. Kuwarananchoren and S. Sundaram, "On the geometric convergence of byzantine-resilient distributed optimization algorithms," 2023, *arXiv:2305.10810*.
[28] K. Kuwarananchoren, L. Xin, and S. Sundaram, "Byzantine-resilient distributed optimization of multi-dimensional functions," in *Proc. Amer. Control Conf.*, 2020, pp. 4399–4404.
[29] H. J. LeBlanc, H. Zhang, X. Koutsoukos, and S. Sundaram, "Resilient asymptotic consensus in robust networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 4, pp. 766–781, Apr. 2013.
[30] A. Sinclair, "Improved bounds for mixing rates of Markov chains and multicommodity flow," *Combinatorics Probab. Comput.*, vol. 1, no. 4, pp. 351–370, 1992.
[31] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.
[32] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Trans. Autom. Control*, vol. 57, no. 3, pp. 592–606, Mar. 2012.
[33] D. Jakovetić, J. Xavier, and J. M. Moura, "Fast distributed gradient methods," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1131–1146, May 2014.
[34] N. A. Lynch, *Distributed Algorithms*. Amsterdam, The Netherlands: Elsevier, 1996.
[35] H. Tverberg, "A generalization of radon's theorem," *J. London Math. Soc.*, vol. 1, no. 1, pp. 123–128, 1966.
[36] J. R. Reay, "An extension of radon's theorem," *Illinois J. Math.*, vol. 12, no. 2, pp. 184–189, 1968.
[37] N. H. Vaidya, "Iterative byzantine vector consensus in incomplete graphs," in *Proc. 15th Int. Conf. Distrib. Comput. Netw.*, 2014, pp. 14–28.
[38] Z. Xiang and N. H. Vaidya, "Brief announcement: Relaxed byzantine vector consensus," in *Proc. 28th ACM Symp. Parallelism Algorithms Architectures*, 2016, pp. 401–403.
[39] H. Park and S. A. Hutchinson, "Fault-tolerant rendezvous of multirobot systems," *IEEE Trans. Robot.*, vol. 33, no. 3, pp. 565–582, Jun. 2017.
[40] W. Abbas, M. Shabbir, J. Li, and X. Koutsoukos, "Interplay between resilience and accuracy in resilient vector consensus in multi-agent networks," in *Proc. IEEE 59th Conf. Decis. Control*, 2020, pp. 3127–3132.
[41] J. Yan, Y. Mo, X. Li, and C. Wen, "A "safe kernel" approach for resilient multi-dimensional consensus," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 2507–2512, 2020.
[42] M. Pirani, A. Mitra, and S. Sundaram, "Graph-theoretic approaches for analyzing the resilience of distributed control systems: A tutorial and survey," vol. 157, Nov. 2023, Art. no. 111264, doi: 10.1016/j.automatica.2023.111264.
[43] D. H. Gutman and J. F. Pena, "The condition number of a function relative to a set," *Math. Program.*, vol. 188, pp. 255–294, 2021.
[44] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87, Berlin, Germany: Springer Science & Business Media, 2003.
[45] L. Guerrero-Bonilla, A. Prorok, and V. Kumar, "Formations for resilient robot teams," *IEEE Robot. Automat. Lett.*, vol. 2, no. 2, pp. 841–848, Apr. 2017.
[46] D. Saldana, A. Prorok, S. Sundaram, M. F. Campos, and V. Kumar, "Resilient consensus for time-varying networks of dynamic agents," in *Proc. Amer. Control Conf.*, 2017, pp. 252–258.
[47] W. Abbas, M. Shabbir, J. Li, and X. Koutsoukos, "Resilient distributed vector consensus using centerpoint," *Automatica*, vol. 136, 2022, Art. no. 110046.
[48] K. Kuwarananchoren and S. Sundaram, "On the set of possible minimizers of a sum of known and unknown functions," in *Proc. Amer. Control Conf.*, 2020, pp. 106–111.
[49] K. Kuwarananchoren and S. Sundaram, "The minimizer of the sum of two strongly convex functions," 2023, *arXiv:2305.13134*.
[50] J. Yan, Y. Mo, X. Li, L. Xing, and C. Wen, "Resilient vector consensus: An event-based approach," in *Proc. IEEE 16th Int. Conf. Control Automat.*, 2020, pp. 889–894.
[51] B. S. Mordukhovich and N. M. Nam, "An easy path to convex analysis and applications," *Synth. Lectures Math. Statist.*, vol. 6, no. 2, pp. 1–218, 2013.
[52] D. Castano, V. E. Paksoy, and F. Zhang, "Angles, triangle inequalities, correlation matrices and metric-preserving and subadditive functions," *Linear Algebra Appl.*, vol. 491, pp. 15–29, 2016.

Kananart Kuwarananchoren (Member, IEEE) received the B.Eng. degree in electrical engineering from Chulalongkorn University, Bangkok, Thailand, in 2016, and the Ph.D. degree from the Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. During his doctoral studies, he was a Research Intern with Intel Labs. He holds the position of AI Research Scientist with Intel Labs. His research interests include distributed optimization and reinforcement learning. He was the recipient of the prestigious University Gold Medal at Chulalongkorn University.

Lei Xin (Student Member, IEEE) received the B.S. degree in electrical and computer engineering (highest distinction) and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2018 and 2023, respectively. He is currently a Postdoctoral Researcher with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. He was a finalist for the Best Student Paper Award at the 2022 American Control Conference. His research interests include machine learning, system identification, and optimization.



Shreyas Sundaram (Senior Member, IEEE) received the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2005 and 2009, respectively. He is currently the Marie Gordon Associate Professor with the Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. From 2009 to 2010, he was a Postdoctoral Researcher with the University of Pennsylvania, Philadelphia, PA, USA, and an Assistant Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, from 2010 to 2014. His research interests include network science, analysis of large-scale dynamical systems, fault-tolerant and secure control, linear system and estimation theory, game theory, and the application of algebraic graph theory to system analysis. He was the recipient of the NSF CAREER Award, and an Air Force Research Lab Summer Faculty Fellowship, Hesselberth Award for Teaching Excellence and Ruth and Joel Spira Outstanding Teacher Award, Department of Electrical and Computer Engineering Research Award at Waterloo, Faculty of Engineering Distinguished Performance Award, M. E. Van Valkenburg Graduate Research Award, and the Robert T. Chien Memorial Award from the University of Illinois. He was a finalist for the Best Student Paper Award at the 2007 and 2008 American Control Conferences.