Approximate Byzantine Fault-Tolerance in Distributed Optimization

Shuo Liu
sl1539@georgetown.edu
Georgetown University
Washington DC, USA

Nirupam Gupta
nirupam.gupta@epfl.ch
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland

Nitin H. Vaidya
nitin.vaidya@georgetown.edu
Georgetown University
Washington DC, USA

ABSTRACT

This paper considers the problem of Byzantine fault-tolerance in distributed multi-agent optimization. In this problem, each agent has a local cost function, and in the fault-free case, the goal is to design a distributed algorithm that allows all the agents to find a minimum point of all the agents' aggregate cost function. We consider a scenario where some agents might be Byzantine faulty that renders the original goal of computing a minimum point of all the agents' aggregate cost vacuous. A more reasonable objective for an algorithm in this scenario is to allow all the non-faulty agents to compute the minimum point of only the non-faulty agents' aggregate cost. Prior work [24] shows that if there are up to f (out of n) Byzantine agents then a minimum point of the non-faulty agents' aggregate cost can be computed *exactly* if and only if the non-faulty agents' costs satisfy a certain redundancy property called $2f$*-redundancy*. However, $2f$-redundancy is an ideal property that can be satisfied only in systems free from noise or uncertainties, which can make the goal of exact fault-tolerance *unachievable* in some applications. Thus, we introduce the notion of (f, ϵ)-resilience, a generalization of exact fault-tolerance wherein the objective is to find an approximate minimum point of the non-faulty aggregate cost, with ϵ accuracy. This approximate fault-tolerance can be achieved under a weaker condition that is easier to satisfy in practice, compared to $2f$-redundancy. We obtain necessary and sufficient conditions for achieving (f, ϵ)-resilience characterizing the correlation between relaxation in redundancy and approximation in resilience. In case when the agents' cost functions are differentiable, we obtain conditions for (f, ϵ)-resilience of the distributed gradient-descent method when equipped with *robust gradient aggregation*; such as *comparative gradient elimination* or *coordinate-wise trimmed mean.*

CCS CONCEPTS

• **Computing methodologies → Distributed algorithms**.

KEYWORDS

Distributed optimization; Approximate fault-tolerance; Distributed gradient-descent

PODC '21, July 26–30, 2021, Virtual Event, Italy

ACM ISBN 978-1-4503-8548-0/21/07…$15.00
https://doi.org/10.1145/3465084.3467902

ACM Reference Format:
Shuo Liu, Nirupam Gupta, and Nitin H. Vaidya. 2021. Approximate Byzantine Fault-Tolerance in Distributed Optimization. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC '21), July 26–30, 2021, Virtual Event, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3465084.3467902

1 INTRODUCTION

The problem of distributed optimization in multi-agent systems has gained significant attention in recent years [8, 17, 33]. In this problem, each agent has a *local cost function* and, when the agents are fault-free, the goal is to design algorithms that allow the agents to collectively minimize the aggregate of their cost functions. To be precise, suppose that there are n agents in the system and let $Q_i(x)$ denote the local cost function of agent i, where x is a d-dimensional vector of real values, i.e., $x \in \mathbb{R}^d$. A traditional distributed optimization algorithm outputs a *global minimum* x^* such that

$$x^* \in \arg\min_{x \in \mathbb{R}^d} \sum_{i=1}^{n} Q_i(x). \tag{1}$$

As a simple example, $Q_i(x)$ may denote the cost for an agent i (which may be a robot or a person) to travel to location x from their current location, and x^* is a location that minimizes the total cost of meeting for all the agents. Such multi-agent optimization is of interest in many practical applications, including distributed machine learning [8], swarm robotics [38], and distributed sensing [37].

We consider the distributed optimization problem in the presence of up to f Byzantine faulty agents, originally introduced by Su and Vaidya [43]. The Byzantine faulty agents may behave arbitrarily [28]. In particular, the non-faulty agents may share arbitrary incorrect and inconsistent information in order to bias the output of a distributed optimization algorithm. For example, consider an application of multi-agent optimization in the case of distributed sensing where the agents (or *sensors*) observe a common *object* in order to collectively identify the object. However, the faulty agents may send arbitrary observations concocted to prevent the non-faulty agents from making the correct identification [12, 14, 35, 44]. Similarly, in the case of distributed learning, which is another application of distributed optimization, the faulty agents may send incorrect information based on *mislabelled* or arbitrary concocted data points to prevent the non-faulty agents from learning a *good* classifier [1, 3, 6, 10, 11, 13, 23, 46].

1.1 Background: Exact Fault-Tolerance

In the *exact fault-tolerance* problem, the goal is to design a distributed algorithm that allows all the non-faulty agents to compute a minimum point of the aggregate cost of only the non-faulty agents [24]. Specifically, suppose that in a given execution, set $\mathcal{B}$ with $|\mathcal{B}| \leq f$ is the set of Byzantine agents, where notation $|\cdot|$ denotes the set cardinality, and $\mathcal{H} = \{1, \ldots, n\} \setminus \mathcal{B}$ denotes the set of non-faulty (i.e., honest) agents. Then, a distributed optimization algorithm has exact fault-tolerance if it outputs a point $x^*_{\mathcal{H}}$ such that

$$x^*_{\mathcal{H}} \in \arg\min_{x \in \mathbb{R}^d} \sum_{i \in \mathcal{H}} Q_i(x). \tag{2}$$

However, since the identity of the Byzantine agents is a priori unknown, in general, exact fault-tolerance is unachievable [43]. Specifically, as shown in [24, 25], exact fault-tolerance can be achieved *if and only if* the agents' cost functions satisfy the $2f$*-redundancy* property defined below.

Definition 1 **($2f$-redundancy).** *The agents' cost functions are said to have* $2f$-redundancy *property if and only if for every pair of subsets S, $\widehat{S} \subseteq \{1, \ldots, n\}$ with $\widehat{S} \subseteq S$, $|S| = n - f$, and $\left|\widehat{S}\right| \geq n - 2f$,*

$$\arg\min_{x \in \mathbb{R}^d} \sum_{i \in \widehat{S}} Q_i(x) = \arg\min_{x \in \mathbb{R}^d} \sum_{i \in S} Q_i(x).$$

In principle, the $2f$-redundancy property can be realized by design for many applications of multi-agent distributed optimization including distributed sensing and distributed learning (see [22, 24]). However, practical realization of $2f$-redundancy can be difficult in the presence of *noise* in the real-world systems. Therefore, we propose a pragmatic generalization of exact fault-tolerance, namely (f, ϵ)*-resilience.*

1.2 (f, ϵ)-Resilience: A Relaxation of Exact Fault-Tolerance

Intuitively, the proposed notion of (f, ϵ)*-resilience* requires an algorithm to output *approximation* of a minimum point of the aggregate of the cost functions of sufficiently large subsets of non-faulty agents. We define (f, ϵ)*-resilience* below, where $\epsilon \in \mathbb{R}_{\geq 0}$ is the measure of approximation and $\|\cdot\|$ denotes the Euclidean norm. The Euclidean distance between a point x and a non-empty set X in space $\mathbb{R}^d$ is denoted by $\text{dist}(x, X)$, and is defined as

$$\text{dist}(x, X) = \inf_{y \in X} \|x - y\|. \tag{3}$$

Definition 2 **((f, ϵ)-resilience).** *A distributed optimization algorithm is said to be (f, ϵ)-resilient if it outputs a point $\widehat{x} \in \mathbb{R}^d$ such that for every subset S of non-faulty agents with $|S| = n - f$,*

$$\text{dist}\left(\widehat{x}, \arg\min_{x \in \mathbb{R}^d} \sum_{i \in S} Q_i(x)\right) \leq \epsilon,$$

despite the presence of up to f Byzantine agents.

Thus, with (f, ϵ)*-resilience*, the output is within distance ϵ of a minimum point of the aggregate cost function of any $n - f$ non-faulty agents. As there can be at most f Byzantine faulty agents whose identity remains unknown, the following two scenarios are indistinguishable in general: (1) there are exactly f Byzantine agents, and (2) there are less than f Byzantine agents. Thus, estimation for the minimum point of the aggregate cost functions of $n - f$ non-faulty agents is indeed a reasonable goal [43]. Analogous resilience requirements have been previously studied in other contexts as well, such as robust statistics (e.g., robust mean estimation [11, 41]) and fault-tolerant linear state estimation [5, 19, 20, 31, 34, 40]. In this work, we address resilience in the context of distributed optimization.

In this paper, we only consider *deterministic* algorithms which, given a fixed set of inputs from the agents, always output the same point in $\mathbb{R}^d$. Thus, a deterministic (f, ϵ)-resilient algorithm produces a unique output point in all of its executions with identical inputs from all the agents (including the faulty ones). **Note that** in the deterministic framework, exact fault-tolerance is equivalent to $(f, 0)$-resilience, i.e., a deterministic $(f, 0)$-resilient algorithm achieves exact fault-tolerance, and vice-versa. Therefore, results on (f, ϵ)-resilience for arbitrary $\epsilon \geq 0$ have a wider application compared to results applicable only to exact fault-tolerance, e.g., [6, 18, 24, 42].

We show that (f, ϵ)*-resilience* requires a *weaker redundancy* condition, in comparison to $2f$-redundancy, named $(2f, \epsilon)$*-redundancy* defined in Definition 3 below. Recall that the *Euclidean Hausdorff distance* between two sets X and Y in $\mathbb{R}^d$, which we denote by $\text{dist}(X, Y)$, is defined as follows [32]:

$$\text{dist}(X, Y) \triangleq \max\left\{\sup_{x \in X} \text{dist}(x, Y), \sup_{y \in Y} \text{dist}(y, X)\right\}. \tag{4}$$

Definition 3 **($(2f, \epsilon)$-redundancy).** *The agents' cost functions are said to have $(2f, \epsilon)$-redundancy property if and only if for every pair of subsets S, $\widehat{S} \subseteq \{1, \ldots, n\}$ with $|S| = n - f$, $\left|\widehat{S}\right| \geq n - 2f$ and $\widehat{S} \subseteq S$,*

$$\text{dist}\left(\arg\min_{x \in \mathbb{R}^d} \sum_{i \in S} Q_i(x), \arg\min_{x \in \mathbb{R}^d} \sum_{i \in \widehat{S}} Q_i(x)\right) \leq \epsilon. \tag{5}$$

It is easy to show that $2f$-redundancy (Definition 1) is equivalent to $(2f, 0)$-redundancy (note that $\epsilon = 0$ here). It is also obvious that $2f$-redundancy implies $(2f, \epsilon)$-redundancy for all $\epsilon \geq 0$. However, the converse need not be true. Thus, the $(2f, \epsilon)$-redundancy property with $\epsilon > 0$ is *weaker* than $2f$-redundancy.

1.3 Applications

Our results are applicable to a large class of distributed optimization problems; including distributed sensing [14, 34, 35, 42], distributed machine learning [7, 8, 13, 48], and distributed linear regression (Section 5). We discuss below the specific case of distributed learning.

Distributed Learning: In this particular optimization problem, each agent has some local *data points* and the goal for the agents is to compute a learning parameter that best models the collective data points observed by all the agents [7]. Specifically, given a learning parameter x, for each data point z, we define a loss function $\ell(x; z)$. Suppose that the data generating distribution of agent i is $\mathcal{D}_i$, and let $\mathbb{E}_{z\sim\mathcal{D}_i}$ denote the expectation with respect to the random data point z over distribution $\mathcal{D}_i$. Then,

$$Q_i(x) \triangleq \mathbb{E}_{z\sim\mathcal{D}_i}\ \ell(x;z)$$

When the distribution of data points is identical for all the agents then the $2f$-redundancy property holds true. However, in practice this is rarely the case [13, 48]. Indeed, different agents may have different data distributions in practice. Therefore, exact fault-tolerance in a pragmatic distributed learning framework is an extremely difficult (if not impossible) goal. In the context of distributed learning, our results on approximate fault-tolerance characterize the relationship between the correlation amongst different agents' data (i.e., degree of redundancy), and the fault-tolerance achieved.

1.4 System architecture

We consider *synchronous* systems. Our results apply to the two architectures shown in Figure 1. In the server-based architecture, the server is assumed to be trustworthy, but up to f agents may be Byzantine faulty. In the peer-to-peer architecture, the agents are connected by a complete network, and up to f of these agents may be Byzantine faulty. Provided that $f < \frac{n}{3}$, an algorithm for the server-based architecture can be simulated in the peer-to-peer system using the well-known *Byzantine broadcast* primitive [30]. For simplicity of presentation, the rest of this paper considers the server-based architecture.

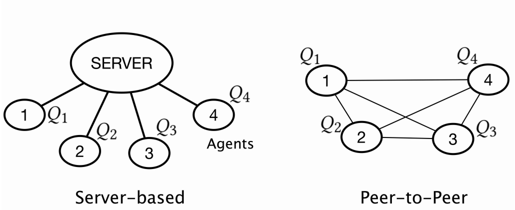


Figure 1: System architecture.

1.5 Summary of Our Contributions

In the **first part** of the paper, i.e., Section 3, we obtain conditions on feasibility and achievability of approximate fault-tolerance of desirable accuracy. Specifically, we show that

- (f, ϵ)-resilience is feasible only if $(2f, \epsilon)$-redundancy property holds true.
- If $(2f, \epsilon)$-redundancy property holds true then $(f, 2\epsilon)$-resilience is achievable.

In the **second part**, i.e., Sections 4 and 5, we consider the case when agents' costs are differentiable, such as in machine learning [7, 47], or regression [22, 42, 44]. We consider the distributed gradient-descent (DGD) method - an iterative distributed optimization algorithm commonly used in this particular case.

- We propose a generic sufficient condition for convergence of the DGD method equipped with a *gradient-filter* (also referred as *robust gradient aggregation*), which is a common fault-tolerance mechanism, e.g., see [6, 13, 24, 48].
- Later, in Section 4.2, we utilize the above result to obtain approximate fault-tolerance properties of the following two specific gradient-filters, under $(2f,\ \epsilon)$-redundancy: (i) Comparative gradient elimination (CGE) [22], and (ii) Coordinate-wise trimmed mean (CWTM) [42]. These two gradient-filters are both easy to implement and versatile [21, 24, 42, 48].
- Finally, in Section 5, we present empirical comparisons between approximate fault-tolerance of the two gradient-filters by simulating a problem of distributed linear regression.

Note: As $(f,\ 0)$-resilience is equivalent to exact fault-tolerance (see Section 1.2), our results on $(f,\ \epsilon)$-resilience encapsulate all the existing results applicable only to exact fault-tolerance, such as the ones in [6, 18, 24, 42].

Compared to related works [26, 27], we present precise redundancy conditions needed for obtaining Byzantine fault-tolerance within a specified approximation error. Unlike them, our results on the impossibility and feasibility of approximate fault-tolerance are applicable to non-differentiable cost functions. Moreover, in the case when the cost functions are differentiable, we present a generic condition for convergence of the DGD method that can precisely model the approximate fault-tolerance property of a generic robust gradient-aggregation rule (a.k.a., gradient-filter).

A full version of this paper including omitted proofs, and additional experimental results and discussion can be found on arXiv [29].

2 OTHER RELATED WORK

In the past, different notions of approximate fault-tolerance, besides $(f,\ \epsilon)$-resilience, have been used to analyze Byzantine fault-tolerance of different distributed optimization algorithms [16, 43]. As we discuss below in Section 2.1, the difference between these other definitions and our definition of $(f,\ \epsilon)$-resilience arises mainly due to the applicability of the distributed optimization problems. Later, in Section 2.2, we discuss some prior work on gradient-filters used for achieving Byzantine fault-tolerance in the distributed gradient-descent method.

2.1 Alternate Notions of Approximation in Fault-Tolerance

As proposed by Su and Vaidya, 2016 [43], instead of a minimum point of the *uniformly weighted* aggregate of non-faulty agents' cost functions, a distributed optimization algorithm may output a minimum point of a *non-uniformly weighted* aggregate of non-faulty costs, i.e., $\sum_{i\in\mathcal{H}} \alpha_i\, Q_i(x)$, where $\mathcal{H}$ denotes the set of at least $n-f$ non-faulty agents, and $\alpha_i \geq 0$ for all $i \in \mathcal{H}$. As is suggested in [43], upon re-scaling the coefficients such that $\sum_{i\in\mathcal{H}} \alpha_i = 1$, we can measure approximation in fault-tolerance using two metrics: (1) the number of coefficients in $\{\alpha_i,\ i \in \mathcal{H}\}$ that are positive, and (2) the minimum positive value amongst the coefficients: $\min\{\alpha_i;\ \alpha_i > 0,\ i \in \mathcal{H}\}$. Results on the achievability of this particular form of approximation for the scalar case (i.e., $d = 1$) can be found in [43, 45]. However, we are unaware of similar results for the case of higher-dimensional optimization problem, i.e., when

$d > 1$. There is some work on this particular notion of approximate fault-tolerance in high-dimensions, such as [42, 47], however their results only apply to special cost functions, specifically, quadratic or strictly convex functions, as opposed to the generic cost functions (that need not even be differentiable) considered in this paper.

Another way of measuring approximation is by the value of the aggregate cost function, or its gradient. For instance, as discussed in [16], for the case of differentiable cost functions a resilient distributed optimization algorithm Π may output a point $x_{\Pi} \in \mathbb{R}^d$ such that each element of the aggregate non-faulty gradient $\sum_{i \in \mathcal{H}} \nabla Q_i(x_{\Pi})$ is bounded by ϵ. As yet another alternative, a resilient algorithm Π may aim to output a point x_{Π} such that the non-faulty aggregate cost $\sum_{i \in \mathcal{H}} Q_i(x_{\Pi})$ is within ϵ of the true minimum cost $\min_x \sum_{i \in \mathcal{H}} Q_i(x)$. However, these definitions of approximate resilience are sensitive to scaling of the cost functions. In particular, if the elements of $\sum_{i \in \mathcal{H}} \nabla Q_i(x_{\Pi})$ are bounded by ϵ then the elements of $\sum_{i \in \mathcal{H}} \alpha \nabla Q_i(x_{\Pi})$ are bounded by $\alpha\epsilon$, where α is a positive scalar value. On the other hand, both $\sum_{i \in \mathcal{H}} Q_i(x)$ and $\sum_{i \in \mathcal{H}} \alpha Q_i(x)$ have identical minimum point regardless of the value of α. Therefore, when the objective is to approximate a minimum point of the non-faulty aggregate cost $\arg\min_x \sum_{i \in \mathcal{H}} Q_i(x)$, which is indeed the case in this paper, use of function (or gradient) values to measure approximation is not a suitable choice.

2.2 Gradient-Filters

In the past, several gradient-filters have been proposed to *robustify* the distributed gradient-descent (DGD) method against Byzantine faulty agents in a server-based architecture, e.g., see [1, 6, 15, 16, 21, 36, 43, 48]. A gradient-filter refers to *Byzantine robust aggregation* of agents' gradients that mitigates the detrimental impact of incorrect gradients sent by the Byzantine agents to the server. To name a few gradient-filters, that are provably effective against Byzantine agents, we have the comparative gradient elimination (CGE) [21, 22], coordinate-wise trimmed mean (CWTM) [43, 48], geometric median-of-means (GMoM) [13], KRUM [6], Bulyan [18], and other spectral gradient-filters [16]. Different gradient-filters guarantee some fault-tolerance under different assumptions on non-faulty agents' cost functions.

In this paper, we propose a generic result, in Theorem 3 in Section 4, on the convergence of the DGD method equipped with a gradient-filter. The result holds true regardless of the gradient-filter used, and thus, can be utilized to obtain formal fault-tolerance property of a gradient-filter in context of the considered distributed optimization problem. We demonstrate this, in Section 4.2, by obtaining (f, ϵ)-resilience properties of two specific gradient-filters; CGE and CWTM. As exact fault-tolerance is equivalent to $(f, 0)$-resilience (see Section 1.2), our results generalize the prior work on exact fault-tolerance of these two filters, see [21, 22, 42]. Moreover, until now, exact fault-tolerance of the CWTM gradient-filter was only studied for special optimization problems of state estimation [42], and machine learning [48]. Our result presents the fault-tolerance property of CWTM for a much larger class of optimization problems.

2.3 Robust Statistics with Arbitrary Outliers

As noted earlier, there has been work on the problem of robust statistics with arbitrary outliers [11, 20, 41]. In this problem, we are given a finite set of data points; α fraction of which are sampled independently and identically from a common distribution $\mathcal{D}$ in $\mathbb{R}^d$, and the remaining $1 - \alpha$ fraction of data points may be arbitrary. The identity of arbitrary data points is a priori unknown, otherwise the problem is trivialized. The objective in this problem is to estimate statistical measures of distribution $\mathcal{D}$, such as mean, or variance, despite the presence of arbitrary outliers. The problem robust mean estimation can potentially be modelled as a fault-tolerant distributed optimization problem where for each non-faulty agent i, $Q_i(x) : (x, x_i) \mapsto y \in \mathbb{R}$ for all $x \in \mathbb{R}^d$ where $x_i \sim \mathcal{D}$. A faulty agent may choose an arbitrary cost function. The cost functions can be designed in a manner such that the minimum point of the aggregate of non-faulty cost functions is equal to the mean for the non-faulty data points. In particular, suppose that for each non-faulty agent i, $Q_i(x) \triangleq \|x - x_i\|^2$ where $x_i \sim \mathcal{D}$. In this case, the minimum point of the non-faulty aggregate cost function is equal to the average of the non-faulty data points sampled from distribution $\mathcal{D}$.

Prior work on robust statistics considers a centralized setting wherein, unlike a distributed setting, all the data points are accessible to a single machine. In this report, we also present distributed algorithms that do not require the agents to share their local data points. Moreover, in the centralized setting, our results are applicable to a larger class of cost functions, including non-convex functions.

2.4 Fault-tolerance in State Estimation

The problem of distributed optimization finds direct application in distributed state estimation [37]. In this problem, the system comprises multiple sensors, and each sensor makes partial observations about the system's state. The goal is to compute the entire state of the system using collective observations from all the sensors. However, if a sensor is faulty then it may share incorrect observations, preventing correct state estimation. The special case of distributed state estimation when the observations are *linear* in the system's state has gained significant attention in the past, e.g. see [5, 14, 31, 34, 35, 39, 40, 42]. These works have shown that the state can be determined despite up to f (out of n) faulty observations *if and only if* the system is $2f$*-sparse observable*, i.e., the complete state can be determined using observations of only $n - 2f$ non-faulty sensors. We note that, in this particular case, $2f$*-sparse observability* is equivalent to $2f$-redundancy. Additionally, some of these works, such as [31, 42], also consider the case of *approximate* linear state estimation when the observations are noisy. Our work is more general in that we consider the problem setting of distributed optimization, and our results apply to a larger class of cost functions.

3 NECESSARY AND SUFFICIENT CONDITIONS FOR (f, ϵ)-RESILIENCE

Throughout this paper we assume, as stated below, that the non-faulty agents' cost functions and their aggregates have well-defined minimum points. Otherwise, the problem of optimization is rendered vacuous.

Assumption 1. *For every non-empty set of non-faulty agents S, we assume that the set $\arg\min_{x \in \mathbb{R}^d} \sum_{i \in S} Q_i(x)$ is non-empty and closed.*

We also assume that $f < n/2$. Lemma 1 below shows that (f, ϵ)-resilience is impossible in general when $f \geq n/2$. Proof of Lemma 1 is easy, and can be found in the full version of this paper on arXiv.

Lemma 1. *If $f \geq n/2$ then there cannot exist a deterministic (f, ϵ)-resilient algorithm for any $\epsilon \geq 0$.*

3.1 Necessary Condition

Theorem 1. *Suppose that Assumption 1 holds true. There exists a deterministic (f, ϵ)-resilient distributed optimization algorithm where $\epsilon \geq 0$* only if *the agents' cost functions satisfy the $(2f, \epsilon)$-redundancy property.*

Proof. To prove the theorem we present a scenario when the agents' cost functions (if non-faulty) are scalar functions, i.e., $d = 1$ and for all i, $Q_i : \mathbb{R} \to \mathbb{R}$, and the minimum point of an aggregate of one or more agents' cost functions is uniquely defined. Obviously, if a condition is necessary in this particular scenario then it is so in the general case involving vector functions with non-unique minimum points.

To prove the necessary condition, we also assume that the server has full knowledge of all the agents' cost functions. This may not hold true in practice, where instead the server may only have partial information about the agents' cost functions. Indeed, this assumption forces the Byzantine faulty agents to a priori fix their cost functions. However, in reality the Byzantine agents may send arbitrary information over time to the server that need not be consistent with a fixed cost function. Thus, necessity of $(2f, \epsilon)$-redundancy under this strong assumption implies its necessity in general.

The proof is by contradiction. Specifically, we show that *If the cost functions of non-faulty agents do not satisfy the $(2f, \epsilon)$-redundancy property then there cannot exist a* deterministic *(f, ϵ)-resilient distributed optimization algorithm.*

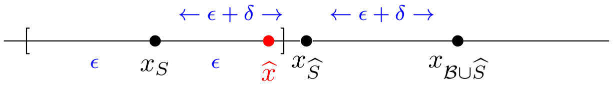
Recall that we have assumed that for a non-empty set of agents T the aggregate cost function $\sum_{i \in T} Q_i(x)$ has a unique minimum point. To be precise, for each non-empty subset of agents T, we define

$$x_T = \arg\min_x \sum_{i \in T} Q_i(x).$$

Suppose that the agents' cost functions **do not** satisfy the $(2f, \epsilon)$-redundancy property stated in Definition 3. Then, there exists a real number $\delta > 0$ and a pair of subsets S, $\widehat{S}$ with $\widehat{S} \subset S$, $|S| = n - f$, and $n - 2f \leq \left|\widehat{S}\right| < n - f$ such that

$$\left\| x_{\widehat{S}} - x_S \right\| \geq \epsilon + \delta. \tag{6}$$

Now, suppose that $n - f - \left|\widehat{S}\right|$ agents in the remainder set $\{1, \ldots, n\} \setminus S$ are Byzantine faulty. Let us denote the set of faulty agents by $\mathcal{B}$. Note that $\mathcal{B}$ is non-empty with $|\mathcal{B}| = n - f - \left|\widehat{S}\right| \leq f$. Similar to the non-faulty agents, the faulty agents send to the server cost functions that are scalar, and the aggregate of one or more agents' cost functions in the set $S \cup \mathcal{B}$ is unique. However, the aggregate cost function of the agents in the set $\mathcal{B} \cup \widehat{S}$ minimizes at a unique point $x_{\mathcal{B} \cup \widehat{S}}$ which is $\left\| x_{\widehat{S}} - x_S \right\|$ distance away from $x_{\widehat{S}}$, similar to x_S, but lies on the other side of $x_{\widehat{S}}$ as shown in the figure below. Note that it is always possible to pick such functions for the faulty agents.



$\leftarrow \epsilon + \delta \rightarrow$
$\leftarrow \epsilon + \delta \rightarrow$
ϵ
x_S
ϵ
$\widehat{x}$
$x_{\widehat{S}}$
$x_{\mathcal{B} \cup \widehat{S}}$

Note that the distance between the two points x_S and $x_{\mathcal{B} \cup \widehat{S}}$ is $2\epsilon + 2\delta$, i.e.,

$$\left\| x_S - x_{\mathcal{B} \cup \widehat{S}} \right\| = 2\epsilon + 2\delta. \tag{7}$$

Now, suppose, toward a contradiction, that there exists an (f, ϵ)-resilient deterministic optimization algorithm named Π. As the identity of Byzantine faulty agents is a priori unknown to the server, and the cost functions sent by the Byzantine faulty agents have similar properties as the non-faulty agents, the server cannot distinguish between the following two possible scenarios; i) S is the set of non-faulty agents, and ii) $\mathcal{B} \cup \widehat{S}$ is the set of non-faulty agents. Note that both the sets S and $\mathcal{B} \cup \widehat{S}$ contain $n - f$ agents.

As the cost functions received by the server are identical in both of the above scenarios, being a deterministic algorithm, Π should have identical output in both the cases. We let $\widehat{x}$ denote the output of Π. In scenario (i) when the set of honest agents is given by S with $|S| = n - f$, as Π is assumed (f, ϵ)-resilient, by Definition 2 the output

$$\widehat{x} \in [x_S - \epsilon, \, x_S + \epsilon] \tag{8}$$

as shown in the figure above. Similarly, in scenario (ii) when the set of honest agents is $\mathcal{B} \cup \widehat{S}$ with $\left|\mathcal{B} \cup \widehat{S}\right| = n - f$,

$$\widehat{x} \in [x_{\mathcal{B} \cup \widehat{S}} - \epsilon, \, x_{\mathcal{B} \cup \widehat{S}} + \epsilon]. \tag{9}$$

However, (7) implies that (8) and (9) cannot be satisfied simultaneously. That is, if Π is (f, ϵ)-resilient in scenario (i) then it cannot be so in scenario (ii), and vice-versa. This contradicts the assumption that Π is (f, ϵ)-resilient. □

3.2 Sufficient Condition

Theorem 2. *Suppose that Assumption 1 holds true. For a real value $\epsilon \geq 0$, if the agents' cost functions satisfy the $(2f, \epsilon)$-redundancy property then $(f, 2\epsilon)$-resilience is achievable.*

Proof. The proof is constructive where we assume that all the agents send their individual cost functions to the server. We assume that $f > 0$ to avoid the trivial case of $f = 0$. Throughout the proof we write the notation $\arg\min_{x \in \mathbb{R}^d}$ simply as $\arg\min$, unless otherwise stated. We begin by presenting an algorithm below, comprising three steps.

Step 1: Each agent sends their cost function to the server. An honest agent sends its actual cost function, while a faulty agent may send an arbitrary function.

Step 2: For each set T of received functions, $|T| = n - f$, the server computes a point

$$x_T \in \arg\min \sum_{i \in T} Q_i(x).$$

For each subset $\widehat{T} \subset T$, $\left|\widehat{T}\right| = n - 2f$, the server computes

$$r_{T\widehat{T}} \triangleq \text{dist}\left(x_T, \ \arg\min \sum_{i \in \widehat{T}} Q_i(x) \right), \tag{10}$$

and

$$r_T = \max_{\substack{\widehat{T} \subset T, \\ \left|\widehat{T}\right| = n-2f}} r_{T\widehat{T}}. \tag{11}$$

Step 3: The server outputs x_S such that

$$S = \arg\min_{\substack{T \subset \{1, \ldots, n\}, \\ |T| = n-f}} r_T. \tag{12}$$

We show that above algorithm is $(f, 2\epsilon)$-resilient under $(2f, \epsilon)$-redundancy. For a non-empty set of agents T, we denote

$$X_T = \arg\min \sum_{i \in T} Q_i(x).$$

Consider an arbitrary set of non-faulty agents G with $|G| = n - f$. Such a set is guaranteed to exist as there are at most f faulty agents, and therefore, at least $n - f$ non-faulty agents exist in the system. Consider an arbitrary set $\widehat{T}$ such that $\widehat{T} \subset G$ and $\left|\widehat{T}\right| = n - 2f$. By Definition 3 of $(2f, \epsilon)$-redundancy,

$$\text{dist}\left(X_G, \ X_{\widehat{T}} \right) \leq \epsilon. \tag{13}$$

Recall from (10) that $r_{G\widehat{T}} = \text{dist}\left(x_G, \ X_{\widehat{T}} \right)$. As $x_G \in X_G$, by Definition (4) of Hausdorff set distance, $\text{dist}\left(x_G, \ X_{\widehat{T}} \right) \leq \text{dist}\left(X_G, \ X_{\widehat{T}} \right)$. Therefore, $r_{G\widehat{T}} \leq \text{dist}\left(X_G, \ X_{\widehat{T}} \right)$, and substituting from (13) implies that

$$r_{G\widehat{T}} \leq \epsilon. \tag{14}$$

Now, recall from (11) that $r_G = \max \left\{ r_{G\widehat{T}} \ \middle| \ \widehat{T} \subset G, \ \left|\widehat{T}\right| = n - 2f \right\}$. As $\widehat{T}$ in (14) is an arbitrary subset of G with $\left|\widehat{T}\right| = n - 2f$,

$$r_G = \max_{\substack{\widehat{T} \subset G, \\ \left|\widehat{T}\right| = n-2f}} r_{G\widehat{T}} \ \leq \ \epsilon. \tag{15}$$

From (12) and (15) we obtain that

$$r_S \leq r_G \leq \epsilon. \tag{16}$$

As $|G| = n - f$, for every set of agents T with $|T| = n - f$, $|T \cap G| \geq n - 2f$. Therefore, for the set S defined in (12), there exists a subset $\widehat{G}$ of G such that $\widehat{G} \subset S$ and $\left|\widehat{G}\right| = n - 2f$. For such a set $\widehat{G}$, by definition of r_S in (11), we obtain that

$$r_{S\widehat{G}} \triangleq \text{dist}\left(x_S, \ X_{\widehat{G}} \right) \leq r_S.$$

Substituting from (16) above, we obtain that

$$\text{dist}\left(x_S, \ X_{\widehat{G}} \right) \leq \epsilon. \tag{17}$$

As $\widehat{G}$ is a subset of G, all the agents in $\widehat{G}$ are non-faulty. Therefore, by Assumption 1, $X_{\widehat{G}}$ is a closed set. Recall that $\text{dist}\left(x_S, \ X_{\widehat{G}} \right) = \inf_{x \in X_{\widehat{G}}} \|x_S - x\|$. The closedness of $X_{\widehat{G}}$ implies that there exists a point $z \in X_{\widehat{G}}$ such that

$$\|x_S - z\| = \inf_{x \in X_{\widehat{G}}} \|x_S - x\| = \text{dist}\left(x_S, \ X_{\widehat{G}} \right).$$

The above, in conjunction with (17), implies that

$$\|x_S - z\| \leq \epsilon. \tag{18}$$

Moreover, as $z \in X_{\widehat{G}}$ where $\widehat{G} \subset G$ with $\left|\widehat{G}\right| = n - 2f$ and $|G| = n - f$, the $(2f, \epsilon)$-redundancy condition stated in Definition 3 implies that $\text{dist}\left(z, \ X_G \right) \leq \epsilon$. Similar to an argument made above, under Assumption 1, X_G is a closed set, and therefore, there exists $x^* \in X_G$ such that

$$\left\| z - x^* \right\| = \text{dist}\left(z, \ X_G \right) \leq \epsilon. \tag{19}$$

By triangle inequality, (18) and (19) implies that $\|x_S - x^*\| \leq \|x_S - z\| + \|z - x^*\| \leq 2\epsilon$. Recall that set G here is an arbitrary set of $n - f$ non-faulty agents. □

It is worth noting that the algorithm constructed in the proof of Theorem 2 only shows sufficiency; it is not a very practical algorithm due to being computationally expensive.

In the next part of the paper, i.e., Sections 4 and 5, we consider the case when the (non-faulty) agents' cost functions are differentiable. Specifically, we study approximate fault-tolerance in the distributed gradient-descent (DGD) method.

4 DISTRIBUTED GRADIENT-DESCENT (DGD) METHOD

In this section, we consider a setting wherein the non-faulty agents' cost functions are differentiable. In this particular case, we study the approximate fault-tolerance of the distributed gradient-descent method coupled with a *gradient-filter*, described below. We consider the server-based system architecture, shown in Fig. 1, assuming a synchronous system.

The DGD method is an iterative algorithm wherein the server maintains an *estimate* of a minimum point, and updates it iteratively using gradients sent by the agents. Specifically, in each iteration $t \in \{0, 1, \ldots\}$, the server starts with an estimate x^t and broadcasts to all the agents. Each non-faulty agent i sends back to the sever the gradient of its cost function at x^t, i.e., $\nabla Q_i(x^t)$. However, Byzantine faulty agents may send arbitrary incorrect vectors as their gradients to the server. The initial estimate, named x^0, is chosen arbitrarily by the server.

A *gradient-filter* is a vector function, denoted by GradFilter, that maps the n gradients received by the server from all the n agents to a d-dimensional vector, i.e., $\mathsf{GradFilter} : \mathbb{R}^{d \times n} \longrightarrow \mathbb{R}^d$. For example, an average of all the gradients as in the case of the traditional distributed gradient-descent method is technically a gradient-filter. However, averaging is not quite robust against Byzantine faulty agents [6, 43]. The real purpose of a gradient-filter is to mitigate the detrimental impact of incorrect gradients sent by the Byzantine faulty agents. In other words, a gradient-filter *robustifies* the traditional gradient-descent method against Byzantine faults. We show that if a gradient-filter satisfies a certain property then it can confer fault-tolerance to the distributed gradient-descent method.

We first formally describe below the steps in each iteration of the distributed gradient-descent method implemented on a synchronous server-based system. Note that we constrain the estimates computed by the server to a compact convex set $\mathcal{W} \subset \mathbb{R}^d$. The set $\mathcal{W}$ can be arbitrarily large. For a vector $x \in \mathbb{R}^d$, its projection onto $\mathcal{W}$, denoted by $[x]_{\mathcal{W}}$, is defined to be

$$[x]_{\mathcal{W}} = \arg\min_{y \in \mathcal{W}} \|x - y\| . \tag{20}$$

As $\mathcal{W}$ is a convex and compact set, $[x]_{\mathcal{W}}$ is unique for each x (see [9]).

4.1 Steps in t-th iteration

In each iteration $t \in \{0, 1, \ldots\}$ the server updates its current estimate x^t to x^{t+1} using Steps S1 and S2 described as follows.

S1: The server requests from each agent the gradient of its local cost function at the current estimate x^t. Each non-faulty agent i will then send to the server the gradient $\nabla Q_i(x^t)$, whereas a faulty agent may send an incorrect arbitrary value for the gradient.

The gradient received by the server from agent i is denoted as g_i^t. If no gradient is received from some agent i, agent i must be faulty (because the system is assumed to be synchronous) – in this case, the server eliminates the agent i from the system, updates the values of n, f, and re-assigns the agents indices from 1 to n.

S2: [Gradient-filtering] The server applies a gradient-filter GradFilter to the n received gradients and computes $\mathsf{GradFilter}\left(g_1^t, \ldots, g_n^t\right) \in \mathbb{R}^d$. Then, the server updates its estimate to

$$x^{t+1} = \left[x^t - \eta_t \, \mathsf{GradFilter}\left(g_1^t, \ldots, g_n^t\right)\right]_{\mathcal{W}} \tag{21}$$

where η_t is the step-size of positive value for iteration t.

We propose, in Theorem 3, a generic convergence result for the above algorithm.

Theorem 3. *Consider the update rule* (21) *in the above iterative algorithm, with diminishing step-sizes* $\{\eta_t, \, t = 0, 1, \ldots\}$ *satisfying* $\sum_{t=0}^{\infty} \eta_t = \infty$ *and* $\sum_{t=0}^{\infty} \eta_t^2 < \infty$*. Suppose that*

$$\left\|\mathsf{GradFilter}\left(g_1^t, \ldots, g_n^t\right)\right\| < \infty$$

for all t. For some point $x^* \in \mathcal{W}$, if there exists real-valued constants $\mathsf{D}^* \in [0, \max_{x \in \mathcal{W}} \|x - x^*\|)$ *and $\xi > 0$ such that for each iteration t,*

$$\phi_t = \left\langle x^t - x^*, \, \mathsf{GradFilter}\left(g_1^t, \ldots, g_n^t\right)\right\rangle \geq \xi \text{ when } \left\|x^t - x^*\right\| \geq \mathsf{D}^*, \tag{22}$$

then $\lim_{t \to \infty} \left\|x^t - x^*\right\| \leq \mathsf{D}^*$.

The values D^* and ξ in Theorem 3 may be interdependent. Proof of Theorem 3 can be found in the full version of this paper on arXiv.

Using Theorem 3 we can obtain conditions under which a gradient-filter guarantees the approximate fault-tolerance property of (f, ϵ)-resilience with $\epsilon \geq 0$, of which exact fault-tolerance is a special case. On the other hand, the prior results on the convergence of DGD method with a gradient-filter, e.g., see [6, 18], apply only to exact fault-tolerance.

We demonstrate below the utility of Theorem 3 to obtain the fault-tolerance properties of two commonly used gradient-filters in the literature; namely *Comparative Gradient Elimination* [21] and *Coordinate-Wise Trimmed Mean* [42].

4.2 Gradient-Filters and their Fault-Tolerance Properties

In this subsection, we present precise approximate fault-tolerance properties of two specific gradient-filters; the Comparative Gradient Elimination (CGE) [21, 22], and the Coordinate-Wise Trimmed Mean (CWTM) [42, 48]. Note that differentiability of non-faulty agents' cost functions, which is already assumed for the DGD method, implies Assumption 1 (see [9]). We additionally make Assumptions 2, 3 and 4 about the non-faulty agents' cost functions. Similar assumptions are made in prior work on fault-free distributed optimization [4, 8, 33].

Assumption 2 (Lipschitz smoothness). *For each non-faulty agent i, we assume that the gradient of its cost function $\nabla Q_i(x)$ is Lipschitz continuous, i.e., there exists a finite real value $\mu > 0$ such that*

$$\left\|\nabla Q_i(x) - \nabla Q_i(x')\right\| \leq \mu \left\|x - x'\right\|, \quad \forall x, x' \in \mathcal{W}.$$

Assumption 3 (Strong convexity). *For a non-empty set of non-faulty agents $\mathcal{H}$, let $Q_{\mathcal{H}}(x)$ denote the average cost function of the agents in $\mathcal{H}$, i.e.,*

$$Q_{\mathcal{H}}(x) = \frac{1}{|\mathcal{H}|} \sum_{i \in \mathcal{H}} Q_i(x).$$

For each such set $\mathcal{H}$ with $|\mathcal{H}| = n - f$, we assume that $Q_{\mathcal{H}}(x)$ is strongly convex, i.e., there exists a finite real value $\gamma > 0$ such that

$$\left\langle \nabla Q(x) - \nabla Q(x'), \, x - x' \right\rangle \geq \gamma \left\|x - x'\right\|^2, \quad \forall x, x' \in \mathcal{W}.$$

Note that, under Assumptions 2 and 3, $\gamma \leq \mu$. The proof of this inequality can be found in the full version of this paper on arXiv. Now, recall that the iterative estimates of the algorithm in Section 4.1 are constrained to a compact convex set $\mathcal{W} \subset \mathbb{R}^d$.

Assumption 4 (Existence). *For each set of non-faulty agents $\mathcal{H}$ with $|\mathcal{H}| = n - f$, we assume that there exists a point* $x_{\mathcal{H}} \in \arg\min_{x \in \mathbb{R}^d} \sum_{i \in \mathcal{H}} Q_i(x)$ *such that* $x_{\mathcal{H}} \in \mathcal{W}$.

We describe below the CGE and CWTM gradient-filters. Later, we obtain the fault-tolerance properties of these filters using the result stated in Theorem 3, under $(2f,\, \epsilon)$-redundancy.

CGE Gradient-Filter: To apply the CGE gradient-filter in Step S2, the server sorts the n gradients received from the n agents at the completion of Step S1 as per their Euclidean norms (ties broken arbitrarily):

$$\left\|g^t_{i_1}\right\| \leq \ldots \leq \left\|g^t_{i_{n-f}}\right\| \leq \left\|g^t_{i_{n-f+1}}\right\| \leq \ldots \leq \left\|g^t_{i_n}\right\|.$$

That is, the gradient with the smallest norm, $g^t_{i_1}$, is received from agent i_1, and the gradient with the largest norm, $g^t_{i_n}$, is received from agent i_n. Then, the output of the CGE gradient-filter is the vector sum of the $n-f$ gradients with smallest $n-f$ Euclidean norms. Specifically,

$$\mathsf{GradFilter}\left(g^t_1, \ldots,\, g^t_n\right) = \sum_{j=1}^{n-f} g^t_{i_j}. \tag{23}$$

CWTM Gradient-Filter: To implement this particular gradient-filter in Step S2, the server sorts the n gradients received from the n agents at the completion of Step S1 as per their individual elements. For a vector $v \in \mathbb{R}^d$, we let $v[k]$ denote its k-th element. Specifically, for each $k \in \{1, \ldots,\, d\}$, the server sorts the k-th elements of the gradients by breaking ties arbitrarily:

$$g^t_{i_1[k]}[k] \leq \ldots \leq g^t_{i_{f+1}[k]}[k] \leq \ldots \leq g^t_{i_{n-f}[k]}[k] \leq \ldots \leq g^t_{i_n[k]}[k].$$

The gradient with the smallest of the k-th element, $g^t_{i_1[k]}$, is received from agent $i_1[k]$, and the gradient with the largest of the k-th element, $g^t_{i_n[k]}$, is received from agent $i_n[k]$. For each k, the server eliminates the largest f and the smallest f of the k-th elements of the gradients received. Then, the output of the CWTM gradient-filter is a vector whose k-th element is equal to the average of the remaining $n-2f$ gradients' k-th elements. That is, for each $k \in \{1, \ldots,\, d\}$,

$$\mathsf{GradFilter}\left(g^t_1, \ldots,\, g^t_n\right)[k] = \frac{1}{n-2f} \sum_{j=f+1}^{n-f} g^t_{i_j[k]}[k]. \tag{24}$$

We present the precise fault-tolerance properties of the two gradient-filters in Theorems 4 and 5 below. However, the **reader may skip to Section 5 without loss of continuity**. Proofs of the theorems can be found in the full version of this paper on arXiv.

Note that, under Assumptions 3 and 4, for each non-empty set of non-faulty agents $\mathcal{H}$ with $|\mathcal{H}| = n-f$, the aggregate cost function $\sum_{i \in \mathcal{H}} Q_i(x)$ has a unique minimum point, denoted by $x_{\mathcal{H}}$, in the set $\mathcal{W}$. Specifically,

$$\{x_{\mathcal{H}}\} = \mathcal{W} \cap \arg\min_{x \in \mathbb{R}^d} \sum_{i \in \mathcal{H}} Q_i(x). \tag{25}$$

We first show below, in Theorem 4, that when the fraction of Byzantine faulty agents f/n is bounded then the DGD method with the CGE gradient-filter is $(f,\, O(\epsilon))$-resilient, under $(2f,\, \epsilon)$-redundancy and the above assumptions.

Theorem 4. *Suppose that the non-faulty agents' cost functions satisfy the $(2f,\, \epsilon)$-redundancy property, and the Assumptions 2, 3 and 4 hold true. Consider the algorithm in Section 4.1 with the CGE gradient-filter defined in (23). The following holds true:*

(1) $\left\|\mathsf{GradFilter}\left(g^t_1, \ldots,\, g^t_n\right)\right\| < \infty$ *for all* t.

(2) If

$$\alpha = 1 - \frac{f}{n}\left(1 + \frac{2\mu}{\gamma}\right) > 0 \tag{26}$$

then for each set of $n-f$ non-faulty agents $\mathcal{H}$, for each $\delta > 0$,

$$\phi_t = \left\langle x^t - x_{\mathcal{H}},\, \mathsf{GradFilter}\left(g^t_1, \ldots, g^t_n\right)\right\rangle \geq \alpha n \gamma \delta \left(\left(\frac{4\mu f}{\alpha\gamma}\right)\epsilon + \delta\right)$$

when $\left\|x^t - x^*\right\| \geq \left(\dfrac{4\mu f}{\alpha\,\gamma}\right)\epsilon + \delta.$

Let $\mathcal{H}$ denote an arbitrary set of $n-f$ non-faulty agents. If the step-size η_t in (21) is diminishing, i.e., $\sum_{t=0}^{\infty}\eta_t = \infty$ and $\sum_{t=0}^{\infty}\eta_t^2 < \infty$, then Theorem 4, in conjunction with Theorem 3 implies that, under the said conditions,

$$\lim_{t\to\infty}\left\|x^t - x_{\mathcal{H}}\right\| \leq \left(\frac{4\mu f}{\alpha\,\gamma}\right)\epsilon + \delta, \quad \forall \delta > 0.$$

The above implies that $\lim_{t\to\infty}\left\|x^t - x_{\mathcal{H}}\right\| \leq (4\mu f/\alpha\gamma)\,\epsilon$. Thus, Theorem 4 shows that under $(2f,\, \epsilon)$-redundancy, and Assumptions 2, 3 and 4, if $\alpha > 0$, or the fraction of Byzantine faulty agents f/n is less than $1/(1+2(\mu/\gamma))$, then the DGD method with the CGE gradient-filter is asymptotically $(f,\, \mathsf{D}\epsilon)$*-resilient* (by Definition 2) where

$$\mathsf{D} = \frac{4\mu f}{\alpha\gamma} = \frac{4\mu\, n}{(n/f)\,\gamma - (\gamma + 2\mu)}. \tag{27}$$

A smaller number f of Byzantine faulty agents implies a smaller value of D, and therefore, better fault-tolerance of the algorithm. Moreover, $\mathsf{D} = 0$ when $f = 0$, i.e., the algorithm indeed converges to the actual minimum point of all the agents' aggregate cost function in the fault-free case. Note that under Assumptions 2 and 3, $\gamma \leq \mu$. So, the fault-tolerance guarantee of the CGE gradient-filter, presented in Theorem 4, requires $f/n < 1/3$, or $f < n/3$.

Next, we show that when the separation between the gradients of the non-faulty agents' cost functions is sufficiently small then the CWTM gradient-filter can guarantee some approximate fault-tolerance under $(2f,\, \epsilon)$-redundancy. To present the fault-tolerance of the CWTM gradient-filter, we make the following additional assumption.

Assumption 5. *For two non-faulty agents i and j, we assume that there exists $\lambda > 0$ such that for all $x \in \mathcal{W}$,*

$$\left\|\nabla Q_i(x) - \nabla Q_j(x)\right\| \leq \lambda \max\left\{\left\|\nabla Q_i(x)\right\|, \left\|\nabla Q_j(x)\right\|\right\}.$$

Due to the triangle triangle inequality, Assumption 5 trivially holds true when $\lambda = 2$. However, we can presently guarantee fault-tolerance of CWTM gradient-filter when $\lambda < \gamma/(\mu\sqrt{d})$ where μ and γ are the Lipschitz smoothness and strong convexity coefficients, respectively defined in Assumption 2 and 3. Recall the definition of point $x_{\mathcal{H}} \in \mathbb{R}^d$ from (25) where $\mathcal{H}$ denotes an arbitrary set of $n-f$ non-faulty agents.

Theorem 5. *Suppose that the non-faulty agents' cost functions satisfy the $(2f,\ \epsilon)$-redundancy property, and the Assumptions 2, 3, 4 and 5 hold true. Consider the algorithm in Section 4.1 with the CWTM gradient-filter defined in (24). The following holds true:*

(1) $\left\| \mathsf{GradFilter}\left(g_1^t, \ldots, g_n^t\right) \right\| < \infty$ *for all* t.

(2) If $\lambda < \gamma/(\mu\sqrt{d})$ *then for each set of* $n - f$ *non-faulty agents* $\mathcal{H}$, *for each* $\delta > 0$,

$$\phi_t = \left\langle x^t - x_{\mathcal{H}},\ \mathsf{GradFilter}\left(g_1^t, \ldots, g_n^t\right) \right\rangle \geq \left(2\sqrt{d}n\mu\lambda\epsilon + \left(\gamma - \sqrt{d}\lambda\mu\right)\delta\right)\delta$$

when $\left\|x^t - x_{\mathcal{H}}\right\| \geq \dfrac{2\sqrt{d}n\mu\lambda}{(\gamma - \sqrt{d}\mu\lambda)}\epsilon + \delta.$

By similar arguments as in the case of CGE, Theorem 5, in conjunction with Theorem 3, implies that the DGD method with CWTM gradient-filter and diminishing step-sizes is asymptotically $(f,\ \mathsf{D}'\,\epsilon)$-*resilient* where

$$\mathsf{D}' = \frac{2\sqrt{d}n\mu\lambda}{(\gamma - \sqrt{d}\mu\lambda)} = \left(\frac{2n}{(\gamma/\mu\lambda\sqrt{d}) - 1}\right),$$

under the conditions stated in Theorem 5. The smaller the value of λ is, i.e., the closer non-faulty gradients to each other are, the smaller is the value of D', and therefore, better is the approximate fault-tolerance guarantee of the CWTM gradient-filter. Unlike the CGE gradient-filter, resilience of CWTM presented in Theorem 5 is independent of f, as long as $\lambda < \gamma/(\mu\sqrt{d})$. However, the condition on λ to guarantee the resilience of CWTM gradient-filter depends upon the dimension d of the optimization problem. Larger dimension result in a tighter bound on λ.

5 NUMERICAL EXPERIMENTS

We present simulation results for an empirical evaluation of the CGE and CWTM gradient-filters applied to the problem of *distributed linear regression* [2]. More experimental details can be found in the full version of this paper on arXiv. We consider the synchronous server-based system in Figure 1. We assume that $n = 6$ and $f = 1$. Each agent i knows a row vector A_i of dimension $d = 2$. Each agent i makes a real-valued (scalar) observation denoted by B_i such that $B_i = A_i x^* + N_i$, where $x^* = (1, 1)^T$ for all i, and N_i is a randomly chosen noise. The value of A_i, B_i and N_i are omitted here for brevity.To solve the linear regression problem distributedly, each agent i's cost function is defined as $Q_i(x) = (B_i - A_i x)^2$. For a non-empty set of agents S, we denote by A_S a matrix of dimension $|S| \times 2$ obtained by stacking rows $\{A_i,\ i \in S\}$. Similarly, we obtain column vector B_S by stacking the values $\{B_i,\ i \in S\}$. Thus for every such non-empty set S, $Q_S(x) \triangleq \sum_{i \in S}(B_i - A_i x)^2 = \|B_S - A_S x\|^2$. The rows $A_1, \ldots, A_n$ are chosen specifically to ensure that the system has $2f$-redundancy if $N_i = 0$ for all i.That is, each matrix A_S with $|S| \geq n - 2f = 4$ is column full-rank or rank $(A_S) = d = 2$. Consequentially, the cost function $Q_S(x)$ has a unique minimum point for each set S with $|S| \geq 4$.

We simulate the distributed gradient-descent algorithm described in Section 4 by assuming agent 1 to be Byzantine faulty, i.e., the set of honest agents is $\mathcal{H} = \{2, 3, 4, 5, 6\}$. The minimum point of $\sum_{i \in \mathcal{H}} Q_i(x)$, denoted by $x_{\mathcal{H}}$, can be obtained by solving $B_{\mathcal{H}} = A_{\mathcal{H}} x$. Specifically, $x_{\mathcal{H}} = (1.0780, 0.9825)^T$. The goal of fault-tolerant distributed linear regression is to estimate $x_{\mathcal{H}}$. In our simulations, it can be verified that the agents' cost functions satisfy the $(2f,\ \epsilon)$-*redundancy* property, stated in Definition 3, with $\epsilon = 0.0890$. It can also be verified that the non-faulty agents' cost functions satisfy Assumptions 2 and 3 with $\mu = 2$ and $\gamma = 0.712$, respectively. We simulate the following fault behaviors for the Byzantine agents:

Table 1: *For the distributed linear regression problem, our algorithm's outputs with gradient-filters CGE and CWTM, and the approximation errors, corresponding to executions when the faulty agent* 1 *exhibits two different types of Byzantine faults;* gradient-reverse *and* random.

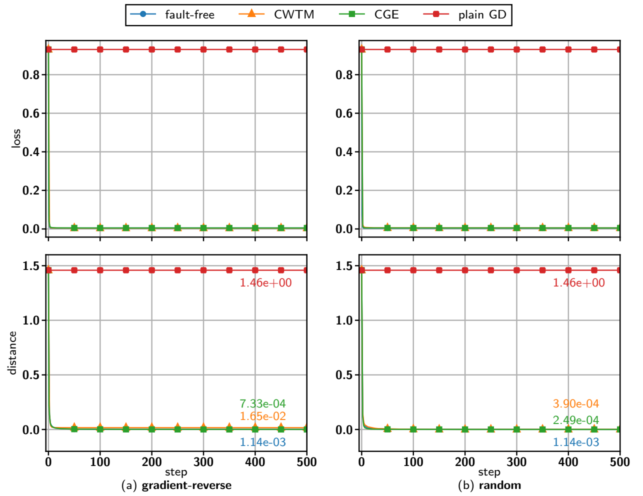
	gradient-reverse		**random**	
	x_{out}	dist $(x_{\mathcal{H}},\ x_{\text{out}})$	x_{out}	dist $(x_{\mathcal{H}},\ x_{\text{out}})$
CGE	$\begin{pmatrix}1.0541\\0.9826\end{pmatrix}$	0.0239	$\begin{pmatrix}1.0779\\0.9826\end{pmatrix}$	4.72×10^{-5}
CWTM	$\begin{pmatrix}1.0645\\0.9924\end{pmatrix}$	0.0167	$\begin{pmatrix}1.0775\\0.9840\end{pmatrix}$	1.51×10^{-3}

- *gradient-reverse*: the faulty agent *reverses* its true gradient. Suppose the correct gradient of a faulty agent i at step t is s_i^t, the agent i will send the incorrect gradient $g_i^t = -s_i^t$ to the server.
- *random*: the faulty agent sends a randomly chosen vector in $\mathbb{R}^d$. In our experiments, the faulty agent in each step chooses i.i.d. Gaussian random vector with mean 0 and an isotropic covariance matrix of standard deviation 200.

In the simulations, we apply a diminishing step size η_t, and a convex compact $\mathcal{W}$ as described in previous sections. For comparison purpose, all experiments have the same initial estimate $x^0 = (-0.0085, -0.5643)^T$. In every execution, the estimates practically converge after 400 iterations. Thus, we document the output of the algorithm to be $x_{\text{out}} = x^{500}$. The outputs for the two gradient-filters, CGE and CWTM, under different faulty behaviors, are shown in Table 1. Note that dist $(x_{\mathcal{H}},\ x_{\text{out}}) = \|x_{\mathcal{H}} - x_{\text{out}}\|$. In all the executions, the distance between $\|x_{\mathcal{H}} - x_{\text{out}}\| < \epsilon$.

For the said executions, we plot in Figure 2 the values of the aggregate cost function $\sum_{i \in \mathcal{H}} Q_i(x^t)$ (referred as *loss*) and the approximation error $\left\|x^t - x_{\mathcal{H}}\right\|$ (referred as *distance*) for iteration t ranging from 0 to 500. We also show the plots of the fault-free DGD method where the faulty agent is omitted, and the DGD method without any gradient-filter when agent 1 is Byzantine faulty. The details for iteration t ranging from 0 to 80 are also highlighted in Figure 3.

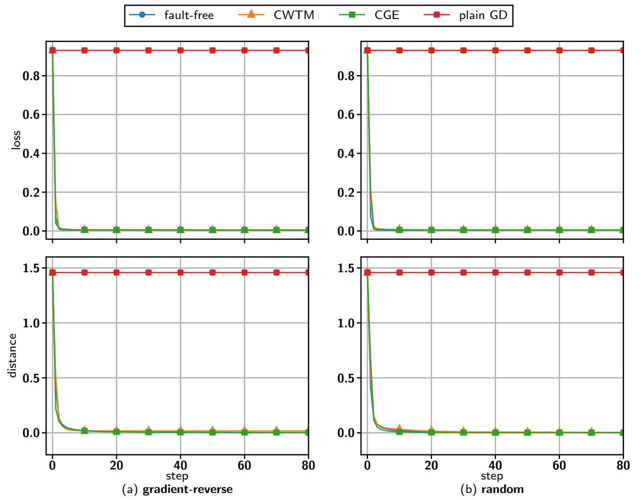
We also conducted experiments for distributed learning with support vector machine with faulty agents in the distributed learning system (see Section 1.3). We observed that the DGD method with the said gradient-filters reaches comparable performance to the fault-free case, and that the accuracy of the learning process depends upon the correlation between the data points of non-faulty agents. For details of those results, please refer to the full version of this paper on arXiv.



(a) **gradient-reverse**

(b) **random**

Figure 2: *The losses, i.e.,* $\sum_{i \in \mathcal{H}} Q_i(x^t)$*, and distances, i.e.,* $\left\|x^t - x_{\mathcal{H}}\right\|$*, versus the number of iterations in the algorithm. The final approximation errors, i.e.,* $\left\|x^{500} - x_{\mathcal{H}}\right\|$*, are annotated in the same colors of their corresponding plots. For the executions shown, agent* 1 *is assumed to be Byzantine faulty. The different columns show the results when the faulty agent exhibits the different types of faults:* (a) *gradient-reverse, and* (b) *random. Apart from the plots with CGE (in* green*) and CWTM (in* yellow*) gradient-filters, we also plot the fault-free DGD method where the faulty agent is omitted (in* blue*), and the DGD method without any gradient-filters when agent* 1 *is Byzantine faulty (in* red*).*



(a) **gradient-reverse**

(b) **random**

Figure 3: *The losses, i.e.,* $\sum_{i \in \mathcal{H}} Q_i(x^t)$*, and distances, i.e.,* $\left\|x^t - x_{\mathcal{H}}\right\|$*, versus the number of iterations in the algorithm, magnified for the initial 80 iterations in the training process. The interpretation of the plots is same as that in Figure 2.*

6 SUMMARY

We have considered the problem of *approximate* Byzantine fault-tolerance – a generalization of the *exact* fault-tolerance problem studied in prior work [24]. Unlike the exact fault-tolerance, the goal in approximate fault-tolerance is to design a distributed optimization algorithm that approximates a minimum point of the aggregate cost function of (at least $n - f$) non-faulty agents, in the presence of up to f (out of n) Byzantine faulty agents. We have defined approximate fault-tolerance formally as (f, ϵ)-resilience where $\epsilon \in \mathbb{R}_{\geq 0}$ represents the approximation error. In the first part of the paper, i.e, Section 3, we have obtained necessary and sufficient conditions for achieving (f, ϵ)-resilience. In the second part of the paper, i.e., Sections 4 and 5, we have considered the case when agents' cost functions are differentiable. In this particular case, we have obtained a generic approximate fault-tolerance property of the distributed gradient-descent method equipped with Byzantine robust gradient aggregation or *gradient-filter*, and have demonstrated the utility of this property by considering two specific well-known gradient-filters; comparative gradient elimination and coordinate-wise trimmed mean. In Section 5, we have demonstrated the applicability of our results through experiments.

ACKNOWLEDGMENTS

Research reported in this paper was supported in part by the Army Research Laboratory under Cooperative Agreement W911NF- 17-2-0196, and by the National Science Foundation award 1842198. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, National Science Foundation or the U.S. Government. Research reported in this paper is also supported in part by a Fritz Fellowship from Georgetown University.

REFERENCES

[1] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. 2018. Byzantine Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/a07c2f3b3b907aaf8436a26c6d77f0a2-Paper.pdf

[2] Takeshi Amemiya. 1985. *Advanced econometrics*. Harvard university press.

[3] Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. 2019. signSGD with Majority Vote is Communication Efficient And Fault Tolerant. arXiv:cs.DC/1810.05291

[4] Dimitri P Bertsekas and John N Tsitsiklis. 1989. *Parallel and distributed computation: numerical methods*. Vol. 23. Prentice hall Englewood Cliffs, NJ.

[5] Kush Bhatia, Prateek Jain, and Purushottam Kar. 2015. Robust Regression via Hard Thresholding. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'15)*. MIT Press, Cambridge, MA, USA, 721–729.

[6] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. 2017. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 118–128.

[7] Léon Bottou, Frank E Curtis, and Jorge Nocedal. 2018. Optimization methods for large-scale machine learning. *Siam Review* 60, 2 (2018), 223–311.

[8] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning* 3, 1 (Jan. 2011), 1–122.

[9] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[10] Xinyang Cao and Lifeng Lai. 2019. Distributed gradient descent algorithm robust to an arbitrary number of byzantine attackers. *IEEE Transactions on Signal Processing* 67, 22 (2019), 5850–5864.

[11] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. 2017. Learning from Untrusted Data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017)*. Association for Computing Machinery, New York, NY, USA, 47–60. https://doi.org/10.1145/3055399.3055491

[12] Yuan Chen, Soummya Kar, and José M. F. Moura. 2018. Resilient Distributed Estimation Through Adversary Detection. *IEEE Transactions on Signal Processing*

66, 9 (2018), 2455–2469.

[13] Yudong Chen, Lili Su, and Jiaming Xu. 2017. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 2 (2017), 44.

[14] Michelle S Chong, Masashi Wakaiki, and Joao P Hespanha. 2015. Observability of linear systems under adversarial attacks. In *American Control Conference*. IEEE, 2439–2444.

[15] Georgios Damaskinos, El Mahdi El Mhamdi, Rachid Guerraoui, Rhicheek Patra, and Mahsa Taziki. 2018. Asynchronous Byzantine Machine Learning (the case of SGD). In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Jennifer Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 1145–1154. http://proceedings.mlr.press/v80/damaskinos18a.html

[16] Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Jacob Steinhardt, and Alistair Stewart. 2019. Sever: A Robust Meta-Algorithm for Stochastic Optimization. arXiv:cs.LG/1803.02815

[17] John C Duchi, Alekh Agarwal, and Martin J Wainwright. 2011. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control* 57, 3 (2011), 592–606.

[18] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. 2018. The Hidden Vulnerability of Distributed Learning in Byzantium. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Jennifer Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 3521–3530. http://proceedings.mlr.press/v80/mhamdi18a.html

[19] Hamza Fawzi, Paulo Tabuada, and Suhas Diggavi. 2014. Secure estimation and control for cyber-physical systems under adversarial attacks. *IEEE Transactions on Automatic control* 59, 6 (2014), 1454–1467.

[20] Jiashi Feng, Huan Xu, and Shie Mannor. 2015. Distributed Robust Learning. arXiv:stat.ML/1409.5937

[21] Nirupam Gupta, Shuo Liu, and Nitin H. Vaidya. 2021. Byzantine Fault-Tolerant Distributed Machine Learning Using Stochastic Gradient Descent (SGD) and Norm-Based Comparative Gradient Elimination (CGE). arXiv:cs.LG/2008.04699

[22] Nirupam Gupta and Nitin H. Vaidya. 2019. Byzantine Fault Tolerant Distributed Linear Regression. arXiv:cs.LG/1903.08752

[23] Nirupam Gupta and Nitin H. Vaidya. 2019. Byzantine Fault-Tolerant Parallelized Stochastic Gradient Descent for Linear Regression. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 415–420. https://doi.org/10.1109/ALLERTON.2019.8919735

[24] Nirupam Gupta and Nitin H. Vaidya. 2020. Fault-Tolerance in Distributed Optimization: The Case of Redundancy. In *Proceedings of the 39th Symposium on Principles of Distributed Computing (PODC '20)*. Association for Computing Machinery, New York, NY, USA, 365–374. https://doi.org/10.1145/3382734.3405748

[25] Nirupam Gupta and Nitin H. Vaidya. 2020. Resilience in Collaborative Optimization: Redundant and Independent Cost Functions. arXiv:cs.DC/2003.09675

[26] Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. 2020. Learning from History for Byzantine Robust Optimization. arXiv:cs.LG/2012.10333

[27] Kananart Kuwaranancharoen, Lei Xin, and Shreyas Sundaram. 2020. Byzantine-resilient distributed optimization of multi-dimensional functions. In *2020 American Control Conference (ACC)*. IEEE, 4399–4404.

[28] Leslie Lamport, Robert Shostak, and Marshall Pease. 2019. *The Byzantine Generals Problem.* Association for Computing Machinery, New York, NY, USA, 203–226.

[29] Shuo Liu, Nirupam Gupta, and Nitin H. Vaidya. 2021. Approximate Byzantine Fault-Tolerance in Distributed Optimization. arXiv:cs.DC/2101.09337

[30] Nancy A Lynch. 1996. *Distributed algorithms.* Elsevier.

[31] Shaunak Mishra, Yasser Shoukry, Nikhil Karamchandani, Suhas N Diggavi, and Paulo Tabuada. 2016. Secure state estimation against sensor attacks in the presence of noise. *IEEE Transactions on Control of Network Systems* 4, 1 (2016), 49–59.

[32] James R Munkres. 2000. *Topology.* Prentice Hall Upper Saddle River, NJ.

[33] Angelia Nedic and Asuman Ozdaglar. 2009. Distributed Subgradient Methods for Multi-Agent Optimization. *IEEE Trans. Automat. Control* 54, 1 (2009), 48–61.

[34] Miroslav Pajic, Insup Lee, and George J Pappas. 2017. Attack-resilient state estimation for noisy dynamical systems. *IEEE Transactions on Control of Network Systems* 4, 1 (2017), 82–92.

[35] Miroslav Pajic, James Weimer, Nicola Bezzo, Paulo Tabuada, Oleg Sokolsky, Insup Lee, and George J Pappas. 2014. Robustness of attack-resilient state estimators. In *ICCPS'14: ACM/IEEE 5th International Conference on Cyber-Physical Systems (with CPS Week 2014)*. IEEE, 163–174.

[36] Adarsh Prasad, Arun Sai Suggala, Sivaraman Balakrishnan, and Pradeep Ravikumar. 2018. Robust Estimation via Robust Gradient Estimation. arXiv:stat.ML/1802.06485

[37] Michael Rabbat and Robert Nowak. 2004. Distributed optimization in sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*. IEEE, 20–27.

[38] Robin L Raffard, Claire J Tomlin, and Stephen P Boyd. 2004. Distributed optimization for cooperative agents: Application to formation flight. In *2004 43rd IEEE Conference on Decision and Control (CDC)*, Vol. 3. IEEE, 2453–2459.

[39] Yasser Shoukry, Pierluigi Nuzzo, Alberto Puggelli, Alberto L Sangiovanni-Vincentelli, Sanjit A Seshia, Mani Srivastava, and Paulo Tabuada. 2015. Imhotep-SMT: A satisfiability modulo theory solver for secure state estimation. In *Proc. Int. Workshop on Satisfiability Modulo Theories.*

[40] Yasser Shoukry, Pierluigi Nuzzo, Alberto Puggelli, Alberto L Sangiovanni-Vincentelli, Sanjit A Seshia, and Paulo Tabuada. 2017. Secure state estimation for cyber-physical systems under sensor attacks: A satisfiability modulo theory approach. *IEEE Trans. Automat. Control* 62, 10 (2017), 4917–4932.

[41] Jacob Steinhardt, Moses Charikar, and Gregory Valiant. 2017. Resilience: A Criterion for Learning in the Presence of Arbitrary Outliers. arXiv:cs.LG/1703.04940

[42] Lili Su and Shahin Shahrampour. 2018. Finite-time Guarantees for Byzantine-Resilient Distributed State Estimation with Noisy Measurements. arXiv:cs.SY/1810.10086

[43] Lili Su and Nitin H. Vaidya. 2016. Fault-Tolerant Multi-Agent Optimization: Optimal Iterative Distributed Algorithms *(PODC '16)*. Association for Computing Machinery, New York, NY, USA, 425–434. https://doi.org/10.1145/2933057.2933105

[44] Lili Su and Nitin H. Vaidya. 2016. Non-Bayesian Learning in the Presence of Byzantine Agents. In *Distributed Computing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 414–427.

[45] Lili Su and Nitin H. Vaidya. 2021. Byzantine-Resilient Multiagent Optimization. *IEEE Trans. Automat. Control* 66, 5 (2021), 2227–2233.

[46] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. 2018. Generalized Byzantine-tolerant SGD. arXiv:cs.DC/1802.10116

[47] Zhixiong Yang and Waheed U. Bajwa. 2017. ByRDiE: Byzantine-resilient distributed coordinate descent for decentralized learning. arXiv:cs.LG/1708.08155

[48] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. 2018. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Jennifer Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 5650–5659. http://proceedings.mlr.press/v80/yin18a.html