Fault-Tolerant Multi-Agent Optimization: Optimal Iterative Distributed Algorithms [*]

Lili Su
Electrical and Computer Engineering
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
lilisu3@illinois.edu

Nitin H. Vaidya
Electrical and Computer Engineering
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
nhv@illinois.edu

ABSTRACT

This paper addresses the problem of distributed multi-agent optimization in which each agent i has a local cost function $h_i(x)$, and the goal is to optimize a global cost function consisting of an average of the local cost functions. Such optimization problems are of interest in many contexts, including distributed machine learning and distributed robotics.

We consider the distributed optimization problem in the presence of faulty agents. We focus primarily on Byzantine failures, but also briefly discuss some results for crash failures. For the Byzantine fault-tolerant optimization problem, the ideal goal is to optimize the average of local cost functions of the non-faulty agents. However, this goal also cannot be achieved. Therefore, we consider a relaxed version of the fault-tolerant optimization problem.

The goal for the relaxed problem is to generate an output that is an optimum of a global cost function formed as a *convex combination* of local cost functions of the non-faulty agents. More precisely, if $\mathcal{N}$ denotes the set of non-faulty agents in a given execution, then there must exist weights α_i for $i \in \mathcal{N}$ such that $\alpha_i \geq 0$ and $\sum_{i \in \mathcal{N}} \alpha_i = 1$, such that the output is an optimum of the cost function $\sum_{i \in \mathcal{N}} \alpha_i h_i(x)$. Ideally, we would like $\alpha_i = \frac{1}{|\mathcal{N}|}$ for all $i \in \mathcal{N}$, however, the maximum number of nonzero weights (α_i's) that can be guaranteed is $|\mathcal{N}| - f$, where f is the maximum number of Byzantine faulty agents.

The contribution of this paper is to present an *iterative* distributed optimization algorithm that achieves optimal fault-tolerance. Specifically, it ensures that at least $|\mathcal{N}| - f$ agents have weights that are bounded away from 0 (in particular, lower bounded by $\frac{1}{2(|\mathcal{N}|-f)}$). The proposed distributed algorithm has a simple iterative structure, with each agent maintaining only a small amount of local state. We show that the iterative algorithm ensures two properties as time goes to ∞: consensus (i.e., output of non-faulty agents becomes identical in the time limit), and optimality (in the sense that the output is the optimum of a suitably defined global cost function). After a finite number of iterations, the algorithm satisfies these properties approximately.

[*]This research is supported in part by National Science Foundation awards NSF 1329681 and 1421918. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government.

PODC'16, July 25–28, 2016, Chicago, IL, USA.
© 2016 ACM. ISBN 978-1-4503-3964-3/16/07…$15.00
DOI: http://dx.doi.org/10.1145/2933057.2933105

Keywords

Distributed optimization; Byzantine faults; complete networks; fault-tolerant computing

1. INTRODUCTION

Distributed optimization over multi-agent networks has received significant attention in recent years [9, 19, 31, 6, 13, 15]. Multi-agent optimization problems are of interest in many contexts, including distributed machine learning and distributed robotics. Most of the prior work in this area assumes that agents are cooperative (in the sense that agents try to follow the pre-specified rules or protocols). In contrast, in this work, we focus on the scenario when some agents may fail.

In the multi-agent optimization problem [9, 19], each agent has a *local* cost function, and when every agent is free from failures, the goal is to design algorithms that allow all the agents collectively minimize the average of the local cost functions. Specifically, let $h_i(x)$ be the local cost function of agent i, with x being the argument, for $i = 1, \cdots, n$. Under some regularity assumptions on the local functions, all the agents cooperatively identify an output x_o such that

$$\text{output } x_o \in \underset{x}{\operatorname{argmin}} \ \frac{1}{n} \sum_{i=1}^{n} h_i(x). \tag{1}$$

This paper explores the problem of distributed optimization in the presence of Byzantine faulty agents. In Section 7, we will also briefly discuss the case when the agents may crash. Byzantine faulty agents can behave arbitrarily. Unless fault-tolerance mechanisms are incorporated, the output of an optimization algorithm can be significantly affected by the faulty agents. In particular, a faulty agent may potentially send incorrect and incosistent messages to the non-faulty agents in order to bias the outcome. When agents may suffer Byzantine faults, instead of optimizing the global cost function defined above, ideally, the agents should attempt optimize the average of the local cost functions of just the non-faulty agents. It should be easy to see that, since the identity of the faulty agents is not known the non-faulty agents, it is not possible to optimize such a global cost function [22, 26]. Therefore, the goal must be relaxed when agents suffer Byzantine failures.

Several ways may be envisioned for relaxing the objective of fault-tolerant optimization. For instance, one way to relax the goal is to allow the output to deviate from the output of the ideal goal above, but to try to minimize the extent of this deviation in the worst-case. In general, we may formulate the relaxed optimization problem to allow a suitably chosen unique output from among the union of the set of optima of the local cost functions of all non-faulty agents. In our work, we are exploring the following relaxation of the fault-tolerant optimization problem [22, 23, 24, 25, 26]. Let $\mathcal{N}$ denote the set of non-faulty agents in a given execution. Let $\boldsymbol{\alpha}$ denote a vector with its i-th element being α_i. We only consider $\boldsymbol{\alpha}$ in which, for $i \notin \mathcal{N}$, $\alpha_i = 0$.

$$\text{output } x_o \text{ such that} \tag{2}$$
$$\text{there } \mathbf{exists} \text{ weight vector } \boldsymbol{\alpha} \text{ for which}$$
$$x_o \in \underset{x}{\operatorname{argmin}} \sum_{i\in\mathcal{N}} \alpha_i h_i(x),$$
$$\sum_{i\in\mathcal{N}} \alpha_i = 1, \ \text{ and } \ \forall i, \ \alpha_i \geq 0$$

That is, the output of all the agents must be equal, and must equal an optimum of a weighted average of the local cost functions. Problem (2) does not require the non-faulty agents to learn the actual weights (α_i's) corresponding to the global cost function that was optimized. Observe that the above relaxation allows the output to be any value from a set of values, where each value potentially corresponds to a different choice of $\boldsymbol{\alpha}$.

To approximate the ideal goal of optimizing the average of local cost functions of just the non-faulty agents (in $\mathcal{N}$), it is desired that the weights (α_i's) above be non-zero for the largest possible number of non-faulty agents, and, preferably, these non-zero weights be as close to $\frac{1}{|\mathcal{N}|}$ as possible. This would ensure that the global cost function that is optimized has approximately equal representation for each agent's cost function. In particular, we define parameters β, γ to characterize the "goodness" of a weight vector $\boldsymbol{\alpha}$ [22, 26].

Definition 1. (β, γ)–**admissibility:** *For $\beta > 0$ and $\gamma \geq 1$, vector $\boldsymbol{\alpha}$ is (β, γ)-admissible if:*
(1) $\sum_{i\in\mathcal{N}} \alpha_i = 1$, $\alpha_i \geq 0$, for each $i \in \mathcal{N}$;
(2) $\alpha_j = 0$ for each $j \notin \mathcal{N}$; and
(3) at least γ elements of $\boldsymbol{\alpha}$ are lower bounded by β.

The following impossibility result is easy to prove [22].

Theorem 1. *[22] In a synchronous system, when up to f agents may be Byzantine faulty, for problem (2), for any $\beta > 0$, it is impossible to guarantee that $\boldsymbol{\alpha}$ is (β, γ)-admissible with $\gamma > |\mathcal{N}| - f$.*

Intuitively, the above theorem says that a convex combination of local objectives may not be admissible when the number of non-zero weights of the convex combination exceeds the threshold $|\mathcal{N}| - f$. This paper presents an iterative distributed algorithm that achieves the *optimal* number of non-zero weights. As discussed in Section 3, we have also designed a more complex algorithm that simulates a centralized solution to the fault-tolerant optimization problem [22, 26].

In its general form, in (2) above, the argument x of the cost function $h_i(x)$ is a k-dimensional vector of reals (i.e., $x \in \mathbb{R}^k$), where $k \geq 1$. In this paper, as a first step towards solving the fault-tolerant multi-agent optimization problem, we consider the special case when $k = 1$, i.e., x is a scalar. Problem (2) remains open for vector arguments with $k \geq 2$. Later in the paper, we discuss the technical difficulty in solving the problem with vector inputs.

The main contribution of this paper is as follows:

- For cost functions with real-valued (scalar) arguments, in a synchronous environment consisting of a completely connected network of $n > 3f$ agents, among which up to f agents may be Byzantine faulty, we present an *iterative distributed* algorithm that is optimal in the sense that it matches the bound in the impossibility result of Theorem 1. In particular, the proposed algorithm achieves $\boldsymbol{\alpha}$ that is $(\frac{1}{2(|\mathcal{N}|-f)}, |\mathcal{N}|-f)$-admissible. The iterative algorithm has a very simple structure, and requires each agent to maintain only a small amount of state. In particular, the algorithm allows a Byzantine faulty agent to send inconsistent messages to different agents.
- The above algorithm can also be extended to a *constrained* version of the optimization problem, to the crash failure model, and to asynchronous environments. While much of the paper addresses Byzantine faults in a synchronous completely connected network, Section 6 and Section 7 summarize the above extensions, as well as a few open problems.

Organization:.

The rest of this paper is organized as follows. Section 2 presents our system model, and assumptions regarding the cost functions. Related work is discussed in Section 3. Section 4 demonstrates a distributed gradient-based algorithm, named SBG, whose performance is analyzed in Section 5. Section 6 and Section 7 summarize extensions and discusse the open problems.

2. SYSTEM MODEL, ASSUMPTIONS AND NOTATIONS

The system under consideration is synchronous, and consists of $n > 3f$ agents, where f is the maximum number of agents that may be Byzantine faulty. The communication network is completely connected (i.e., each agent has a communication channel to each of the agents). We discuss some extensions of our results (e.g., for the crash failure model) in Sections 6 and 7. The set of n agents is denoted $\mathcal{V} = \{1, \cdots, n\}$. In a given execution, let $\mathcal{F}$ denote the set of Byzantine faulty agents, and let $\mathcal{N} = \mathcal{V} - \mathcal{F}$ denote the set of non-faulty agents. The set $\mathcal{F}$ of faulty agents may be chosen by an adversary arbitrarily, and may be different across executions.

We say that a function $h(\cdot) : \mathbb{R} \to \mathbb{R}$ is *admissible* if (i) $h(\cdot)$ is convex, and continuously differentiable, (ii) the set $\arg\min_{x\in\mathbb{R}} h(x)$ containing the optima of $h(\cdot)$ is non-empty and compact (i.e., bounded and closed), (iii) the magnitude of the gradients are bounded by L, i.e., $|h'(x)| \leq L$, and the derivatives $h'(\cdot)$ are L–Lipschitz continuous. Each agent $i \in \mathcal{V}$ is initially provided with an *admissible* local cost function $h_i(\cdot) : \mathbb{R} \to \mathbb{R}$. Similar assumptions on the local functions are standard in past literature on failure-free distributed optimization [9, 19, 31, 6, 13, 15].

3. RELATED WORK

The distributed optimization problem is related to Byzantine fault-tolerant consensus. In particular, the non-faulty agents are all required to produce (approximately) equal output, thus, consensus is part of the requirements satisfied by any solution for our problem. There is a significant body of work on fault-tolerant consensus, including our own prior work [7, 5, 18, 10, 14, 36, 11, 34, 30, 29, 12]. While much of the past work on Byzantine consensus has considered scalar inputs, there is recent work that addresses Byzantine consensus with vector (or multidimensional) inputs [17, 35].

Although convex optimization, including distributed optimization, also has a long history, we believe our work is the first to explore the problem of designing Byzantine fault-tolerant algorithms that achieve optimality in the sense defined earlier. Primal and dual decomposition methods that lend themselves naturally to a distributed paradigm have been known for at least fifty years, and their behavior is well understood [3]. The seminal work of Tsitsiklis and colleagues [33, 32] analyze algorithms for minimization of a smooth function $h(x)$ by distributing the processing of the components vector $x \in \mathbb{R}^n$ among n agents assuming $h(x)$ is separable. As noted earlier, there has been significant research on problem (1). The need for robustness for distributed optimization problems has received some attentions recently [9, 15, 28, 22]. Duchi et al. [9] and Lobel and Ozdaglar [15] study the impact of random communication link failures on the convergence of distributed variant of dual averaging algorithm and sub-gradient method, respectively. In particular, both [9] and [15] assume that each realizable link failure pattern admits a doubly-stochastic matrix which governs local estimates evolution dynamics.

Byzantine agents are first considered in the context of optimization in our series of four technical reports [22, 23, 24, 25]. In particular, in [26, 22], we explore an algorithm that requires all agents to perform *reliable broadcast* of all messages, essentially preventing the faulty agents from sending inconsistent messages to non-faulty agents. This approach effectively reduces the distributed optimization problem to a centralized problem, and allows a different class of solutions, with somewhat different properties, and requiring other proof techniques. In [23], we explore the special case when the optimum sets of the local cost functions of the non-faulty agents have a non-empty intersection. This property provides a form of redundnacy, which is exploited to solve the fault-tolerant optimization problem. In contrast, the algorithm presented in this paper is fully distributed, with a simple iterative structure (not requiring *reliable broadcast* and not making any assumptions about the intersection of the optimum sets).

Subsequent to our work, a somewhat weaker fault model (called *malicious* faults) was considered in [28] – they also obtain weaker guarantees on the behavior of their algorithm compared to those obtained in our work (including this paper).

4. BYZANTINE GRADIENT METHOD

A large number of *failure-free* optimization algorithms used in practice are *iterative* in nature, due to the practical benefits of implementing such algorithms [4, 2]. In an iterative algorithm, each agent maintains an estimate of the optimum, and the estimate converges to a true optimum in the limit as the number of iterations $\to \infty$. In this section, we present an iterative algorithm for problem (2). The algorithm satisfies the requirement in (2) in the limit as the number of iterations $\to \infty$. In practice, since the algorithm is terminated after a finite number of iterations, the algorithm will satisfy the requirements *approximately*. In particular, our algorithm will ensure that, after a sufficiently large number of iterations, the estimates maintained by the non-faulty agents become approximately equal (i.e., within some desired $\epsilon_1 > 0$), and the estimate of each agent is also approximately equal to (i.e., within some desired $\epsilon_2 > 0$) the optimum of a suitable weighted cost function (i.e., argmin in (2)). For brevity, our presentation below focusses on the limiting behavior as time $\to \infty$, although our algorithm does provably exhibit the desired behavior as above (i.e., approximate equality) after adequately large finite number of iterations.

The proposed iterative algorithm, *synchronous Byzantine gradient* method (SBG), is presented below. The pseudocode describes the steps that should be performed by each agent $j \in \mathcal{V}$. A Byzantine faulty agent may deviate from the specification arbitrarily. Algorithm SBG combines features of iterative Byzantine consensus algorithms [16, 36] with elements of gradient-based optimization [4, 20]. We will show that the SBG algorithm solves (2) with $\left(\frac{1}{2(|\mathcal{N}|-f)},\, |\mathcal{N}| - f\right)$–admissible weight vector $\boldsymbol{\alpha}$.

Algorithm SBG uses a trimming function (defined below) analogous to that used in previous Byzantine consensus algorithms.

Trim ($\mathcal{D}$)

Input: Multi-set $\mathcal{D}$ of real-valued scalars, with $|\mathcal{D}| \geq 2f+1$.

Output: Sort the elements in $\mathcal{D}$ in a non-decreasing order (breaking ties arbitrarily), and remove the smallest f values and the largest f values. Denote the minimum and the maximum of the remaining $|\mathcal{D}| - 2f$ values as y_s and y_l, respectively. Return the value below:

$$\frac{1}{2}\left(y_s + y_l\right)$$

Each agent j maintains a state variable x_j. We denote the value of x_j at the end of t iterations of SBG as $x_j[t]$, with the initial value being $x_j[0]$. The initial value may be chosen by each agent arbitrarily. Let $h_j'\left(x_j[t-1]\right)$ denote the gradient of agent j's local cost function $h_j(\cdot)$ at $x_j[t-1]$.

$\lambda[t-1]$ used in Step 3 of the algorithm below is called the *step size*, which depends on the iteration index. The step sizes are known to all agents as a priori, and satisfy the following constraints: $\lambda[t-1] \geq \lambda[t]$ for $t \geq 1$, $\sum_{t=1}^{\infty} \lambda[t-1] = \infty$ and $\sum_{t=1}^{\infty} \lambda^2[t-1] < \infty$.

Algorithm **SBG** for agent j in iteration $t \geq 1$:

Step 1: Send the 2-tuple $\left(x_j[t-1], h_j'\left(x_j[t-1]\right)\right)$ to all the other agents.

Step 2: Receive 2-tuples from all the other agents, with the first element of each tuple being a *state variable*, and the second element being a *gradient*. If such a tuple is not received from some agent, assume a default value for the tuple. Define:

$\mathcal{D}_j^x[t-1] \triangleq$ multi-set containing $x_j[t-1]$ and state variables received from other agents;

$\mathcal{D}_j^g[t-1] \triangleq$ multi-set containing $h'_j\left(x_j[t-1]\right)$ and gradients received from other agents.

Step 3: Define $\widetilde{x}_j[t-1] \triangleq \mathsf{Trim}(\mathcal{D}_j^x[t-1])$, and $\widetilde{g}_j[t-1] \triangleq \mathsf{Trim}(\mathcal{D}_j^g[t-1])$. Update state as follows.

$$x_j[t] = \widetilde{x}_j[t-1] - \lambda[t-1]\widetilde{g}_j[t-1]. \tag{3}$$

In Algorithm SBG, in each iteration, an agent exchange both local estimate and local gradient with other agents. Although the information of gradient is contained in the local estimate, by exchanging gradients directly, a desired structural property of the update can be guaranteed. As seen above, each agent j maintains minimal state (namely, x_j) across iterations. Since the Trim function is applied to the state variables and gradients separately, it is possible that the values received from different sets of agents are removed in each of those trimming operations. The key difficulty in proving the desired (β, γ)-admissibility result arises due to the possibility that the Byzantine agents send different (erroneous) gradients to different non-faulty agents. Unlike the failure-free version of distributed optimization, the Byzantine faulty agents can effectively tamper with the global cost function being optimized. Thus, proving the lower bounds on β and γ requires us to show that the impact of the faulty behavior can be bounded. To delineate the impact of the faulty behavior, we now define a family $\mathcal{C}$ of "valid" global cost functions.

$$\mathcal{C} \triangleq \left\{ p(x): \; p(x) = \sum_{i\in\mathcal{N}} \alpha_i h_i(x), \right.$$
$$\left. \text{where } \boldsymbol{\alpha} \text{ is } \left(\tfrac{1}{2(|\mathcal{N}|-f)}, |\mathcal{N}|-f\right)\text{–admissible} \right\} \tag{4}$$

Each $p(x) \in \mathcal{C}$ is said to be a *valid* function (a valid global objective). Note that each $p(x) \in \mathcal{C}$ is a convex combination of local cost functions of the non-faulty agents in $\mathcal{N}$ with $(\frac{1}{2(|\mathcal{N}|-f)}, |\mathcal{N}|-f)$–admissible weight vector $\boldsymbol{\alpha}$.

Time-varying step sizes, as in Step 3 above, are also used in previous algorithms for distributed optimization [9, 19, 31]. The key difference from the prior distributed optimization algorithms is in the use of the trimming function in Step 3 above. As seen later in Lemma 2, despite the adversarial behavior of the faulty agents, it is guaranteed that the *effective gradient* $\widetilde{g}_j[t-1]$ obtained in Step 3 has an important correspondence to a *time-dependent* (i.e., varying with iteration index t) *valid* cost function in $\mathcal{C}$.

Now let us define set Y to be the union of optimal solutions for all the valid functions in $\mathcal{C}$, i.e.,

$$Y \triangleq \bigcup_{p(\cdot)\in\mathcal{C}} \operatorname*{argmin}_{x\in\mathbb{R}} \; p(x). \tag{5}$$

Lemma 1 identifies an important property of set Y that is crucial in our convergence analysis. We will show that, as $t \to \infty$, for each agent j, its state variable $x_j[t]$ becomes trapped in set Y.

Lemma 1. *Set Y is convex and closed.*

Lemma 1 is proved in Appendix A. As stated in Section 2, the argument x of the cost functions $h_j(x)$ is assumed to be a scalar in $\mathbb{R}$. In general, we would like to allow a vector argument for the cost functions (i.e., $x \in \mathbb{R}^k$, $k \geq 2$). However, set Y analogously defined for the case of vector arguments is *not necessarily convex*, making it difficult to extend our proof technique to vector arguments. In fact, the case of vector arguments remains an open problem presently.

The following distance metric will help state the main claim of this paper.

Definition 2. *For any $x \in \mathbb{R}$, the distance between x and set Y is defined as follows*

$$Dist\,(x, Y) \triangleq \inf_{y\in Y} |x - y|\,.$$

Since Y is convex (Lemma 1), the function $Dist\,(\cdot, Y)$ is also convex. Theorem 2 states our main result, which summarizes the convergence behavior of algorithm SBG.

Theorem 2. *Algorithm SBG achieves*

- *Consensus:* $\lim_{t\to\infty} \; (x_i[t] - x_j[t]) = 0$ *for all $i, j \in \mathcal{N}$, and*
- *Optimality:* $\lim_{t\to\infty} \; Dist\,(x_i[t], Y) = 0$ *for each $i \in \mathcal{N}$.*

Interpretation of Theorem 2.

Consensus: Property (i) in the above theorem implies consensus, since the state variables of all non-faulty agents become identical in the limit. However, property (i) does not imply that $x_j[t]$ for each $j \in \mathcal{N}$ itself has a limit. In fact, the value of the state variable $x_j[t]$ may change with t indefinitely. However, as property (i) states, in the limiting behavior, the state variables of all the non-faulty agents change in unison, maintaining consensus. This lack of a limit for each individual $x_j[t]$ is a direct result of the simple structure of algorithm SBG, which allows a faulty agent to send different gradients to different agents. If we were to require a *Byzantine broadcast* of $(x_j[t-1], h'_j(x_j[t-1]))$ in Step 1 of algorithm SBG at each agent j, then such duplicitous behavior by faulty agents can be precluded, as we have shown elsewhere [26]. With a higher cost (due to use of Byzantine broadcast), the modified algorithm can ensure that $x_j[t]$, $j \in \mathcal{N}$ has a limit as $t \to \infty$. However, the use of Byzantine broadcast essentially reduces the distributed optimization problem to the problem of centralized fault-tolerant optimization [26].

Despite the fact that the limit of $x_j[t]$ may not exist, the property (i) is useful in practice – if we were to terminate the algorithm after a sufficiently large number of iterations, then property (i) guarantees that the state of all non-faulty agents will be close to each other, thus achieving approximate consensus.

Optimality: Property (ii) in the above theorem makes guarantees about the "goodness" of the state of non-faulty agents as $t \to \infty$. In particular, observe that $Dist\,(x, Y) = 0$ if and only if $x \in Y$. Thus, property (ii) guarantees that, for sufficiently large t, state $x_j[t]$ for any non-faulty agent j approximately equals an optimum of a *valid* function in $\mathcal{C}$. That is, $x_j[t]$ approximately equals a solution of problem (2) with a $\left(\frac{1}{2(|\mathcal{N}|-f)}, |\mathcal{N}|-f\right)$–admissible weight vector $\boldsymbol{\alpha}$. Thus, Theorems 1 and 2 together imply that algorithm SBG achieves *optimal fault-tolerance* in the sense that an optimal number (i.e., $|\mathcal{N}|-f$) of local cost functions of non-faulty agents are guaranteed to be represented in the global cost function that is optimized. However, as the discussion of property (i) would suggest, this global cost function is time-varying. Secondly, when $|\mathcal{N}|-f$ weights are non-zero, if the weight distribution were to be uniform, then each weight would be $\frac{1}{|\mathcal{N}|-f}$. The fact that the $\boldsymbol{\alpha}$ vector achieved by algorithm SBG is $\left(\frac{1}{2(|\mathcal{N}|-f)}, |\mathcal{N}|-f\right)$–admissible implies that at least $|\mathcal{N}|-f$ weights are $\geq \frac{1}{2(|\mathcal{N}|-f)}$, which is within a factor of 2 of the uniform weight $\frac{1}{|\mathcal{N}|-f}$.

5. CORRECTNESS OF THEOREM 2

In this section, we present some key results that are useful in proving Theorem 2.

Recall that in Step 3 of algorithm SBG, each agent j applies the trimming function to compute the *effective gradient* $\widetilde{g}_j[t-1]$. Lemma 2 establishes a correspondence between this effective gradient and a valid function in $\mathcal{C}$.

LEMMA 2. *For each non-faulty agent $j \in \mathcal{N}$ and each iteration $t \geq 1$, there exists a valid function*

$$p_t^j(x) = \sum_{i\in\mathcal{N}} b_{ji}[t]\, h_i(x) \in \mathcal{C}$$

such that the effective gradient $\widetilde{g}_j[t-1]$ computed in Step 3 of algorithm SBG can be expressed as

$$\widetilde{g}_j[t-1] = \sum_{i\in\mathcal{N}} b_{ji}[t]\, h_i'(x_i[t-1]). \tag{6}$$

The proof of Lemma 2 can be found in [27, 24].

For agent $j \in \mathcal{N}$, the *weights* ($b_{ji}[t]$'s) in the interpolation on the right side of (6) correspond to the weights used to obtain a valid global cost function $p_t^j(x) \in \mathcal{C}$. Note that the vector formed by weights $b_{ji}[t]$ is $\left(\frac{1}{2(|\mathcal{N}|-f)}, |\mathcal{N}|-f\right)$–admissible. It is also important to note that, $h_i'(x_i[t-1])$ for each $i \in \mathcal{N}$ used in (6) is the gradient of agent i's local cost function $h_i(\cdot)$ computed at agent i's *own* state variable $x_i[t-1]$. Thus, the effective gradient $\widetilde{g}_j[t-1]$ is a linear interpolation of gradients of local cost functions at potentially *different* argument values. Despite this apparent discrepancy, algorithm SBG approximates the behavior of a gradient-based distributed optimization algorithm. Intuitively, the reason for this behavior is that the agents are guaranteed to eventually arrive at a consensus – thus, eventually, the gradients at different non-faulty agents are computed at approximately equal arguments. However, the weights $b_{ji}[t]$ in Lemma 2 are time-dependent, due to the potentially incorrect behavior by faulty agents (i.e., the weights correspond to potentially different functions in $\mathcal{C}$ in different iterations). More importantly, at different agents in $\mathcal{N}$, the weights corresponding to the effective gradients in a given iteration t can be different (i.e., for two different agents $k, j \in \mathcal{N}$, $p_t^k(x)$ and $p_t^j(x)$ may be different). Despite this difference, consensus is achieved due to the fact that set Y is convex (Lemma 1) and decreasing step sizes $\lambda[t-1]$ are used in algorithm SBG.

By a similar argument as in proving Lemma 2, we can obtain the result below for $\widetilde{x}_j[t-1]$ computed in Step 3 of algorithm SBG.

COROLLARY 1. *For each non-faulty agent $j \in \mathcal{N}$ and $t \geq 1$, there exists a $\left(\frac{1}{2(|\mathcal{N}|-f)}, |\mathcal{N}|-f\right)$–admissible weight vector $\mathbf{a}_j[t]$ (whose i-th element is $a_{ji}[t]$) such that $\widetilde{x}_j[t-1]$ computed in Step 3 of algorithm SBG can be expressed as*

$$\widetilde{x}_j[t-1] = \sum_{i\in\mathcal{N}} a_{ji}[t]\, x_i[t-1]. \tag{7}$$

Note that weights $b_{ji}[t]$ and $a_{ji}[t]$ in (6) and (7), respectively, are not necessarily identical, because the state variables and gradients are trimmed independently in Step 3 of algorithm SBG.

In Sections 5.1 and 5.2, we sketch the proofs of properties (i) and (ii) stated in Theorem 2. For complete proofs, please see [27, 24].

5.1 Asymptotic Consensus

We now sketch the proof that asymptotic consensus among the non-faulty agents is achieved under algorithm SBG. A complete proof can be found in the full version. Recall that $\lambda[t] \leq \lambda[t-1]$ and $\lim_{t\to\infty} \lambda[t] = 0$. The following proposition is used in proving consensus.

PROPOSITION 1. *Let $0 \leq b < 1$, and $\ell(t) = \sum_{r=0}^{t-1} \lambda[r] b^{t-r}$. Then $\lim_{t\to\infty} \ell(t) = 0$. Additionally, if $\lambda[t] = \frac{1}{t}$ for $t \geq 1$ and $\lambda[0] = 1$* [1]*, then $\ell(t) = O(\frac{1}{t})$.*

Denote $M(t) = \max_{i\in\mathcal{N}} x_i[t]$ and $m(t) = \min_{i\in\mathcal{N}} x_i[t]$.

LEMMA 3. *Under algorithm SBG, $\lim_{t\to\infty} (M[t] - m[t]) = 0$. Additionally, if $\lambda[t] = \frac{1}{t}$ for $t \geq 1$ and $\lambda[0] = 1$, then $(M[t] - m[t]) = O(\frac{1}{t})$.*

From Corollary 1, we know that $\mathbf{a}_j$ is $\left(\frac{1}{2(|\mathcal{N}|-f)}, |\mathcal{N}|-f\right)$–admissible. Since $(|\mathcal{N}|-f) + (|\mathcal{N}|-f) - |\mathcal{N}| \geq 1$, for any $j, k \in \mathcal{N}$, there exists $i^* \in \mathcal{N}$ such that both $a_{ji^*}[t]$ and $a_{ki^*}[t]$ are lower bounded by $\frac{1}{2(|\mathcal{N}|-f)}$. Then the following holds.

$$\begin{aligned} &|\widetilde{x}_j[t-1] - \widetilde{x}_i[t-1]| \\ &\leq \left(1 - \frac{1}{2\left(|\mathcal{N}|-f\right)}\right)\left(M[t-1] - m[t-1]\right). \end{aligned} \tag{8}$$

Similarly, we have

$$|\widetilde{g}_j[t-1] - \widetilde{g}_k[t-1]| \leq \left(1 - \frac{1}{2\left(|\mathcal{N}|-f\right)}\right) 2L, \tag{9}$$

recalling that $|h_k'(x)| \leq L$ for any $k \in \mathcal{N}$ and any $x \in \mathbb{R}$.

From (3), (8) and (9), the disagreement $(M[t] - m[t])$ can

[1]As it can be seen from the proof of Proposition 1, $\lambda[0]$ can be chosen to be any positive constant.

be bounded as follows.

$$\begin{aligned}M[t]-m[t] &\le \left(1-\frac{1}{2\left(|\mathcal{N}|-f\right)}\right)\left(M[t-1]-m[t-1]\right)\\&\quad+2L\lambda[t-1]\left(1-\frac{1}{2\left(|\mathcal{N}|-f\right)}\right)\\&\le \left(1-\frac{1}{2\left(|\mathcal{N}|-f\right)}\right)^{t}\left(M[0]-m[0]\right)\\&\quad+2L\sum_{r=0}^{t-1}\lambda[r]\left(1-\frac{1}{2\left(|\mathcal{N}|-f\right)}\right)^{t-r}. \qquad (10)\end{aligned}$$

The first term in the right hand side of (10) diminishes exponentially fast, and by Proposition 1, the second term also goes to 0. Our next lemma (Lemma 4) says that the cumulative disagreement between two non-faulty agents is finite.

LEMMA 4. *Under algorithm SBG, it holds that*

$$\sum_{t=0}^{\infty}\lambda[t]\left(M[t]-m[t]\right)<\infty.$$

The above results establish asymptotic consensus behavior of SBG.

5.2 Convergence Analysis

In this section, we sketch the proof that $x_j[t]$, where $j\in\mathcal{N}$, is trapped in Y asymptotically.

Lemma 3 shows that $\lim_{t\to\infty}\left(x_j[t]-x_i[t]\right)=0$ for any $j,i\in\mathcal{N}$ – asymptotic consensus among the non-faulty agents is achieved. Thus, for sufficiently large t, (3) approximately equals

$$x_j[t] \;\approx\; x_j[t-1]-\lambda[t-1]p_t^j\left(x_j[t-1]\right), \forall\, j\in\mathcal{N}, \qquad (11)$$

where $p_t^j(\cdot)$ is the valid function identified in Lemma 2. For a non-faulty agent j, one typical trajectory of x_j is as follows: (1) x_j first approaches set Y (defined in (5)), and (2) then bounces back and forth around set Y before completely being trapped in Y. The existence of disagreement in finite time (discussed above) complicates the convergence analysis significantly. To show the optimality of x_j for $j\in\mathcal{N}$, we use the auxiliary sequence $\{z[t]\}_{t=0}^{\infty}$ defined as follows.

DEFINITION 3. *Let $\{z[t]\}_{t=0}^{\infty}$ be a sequence of estimates such that*

$$z[t]=x_{j_t}[t], \quad \textit{where } j_t\in \mathrm{argmax}_{j\in\mathcal{N}} Dist\left(x_j[t],Y\right). \qquad (12)$$

To show $\lim_{t\to\infty} Dist\left(x_j[t],Y\right)=0\,\forall\, j\in\mathcal{N}$, it is enough to show $\lim_{t\to\infty} Dist\left(z[t],Y\right)=0$, observing that $Dist\left(x_j[t],Y\right)\le Dist\left(z[t],Y\right)$ for each $j\in\mathcal{N}$ (formally proved in the full version). Moreover, a simple iteration relation can be obtained for $\{Dist\left(z[t],Y\right)\}_{t=0}^{\infty}$. Specifically,

$$\begin{aligned}Dist\left(z[t+1],Y\right)&\le \max\{\lambda[t]L,\; Dist\left(z[t],Y\right)\}\\&\quad+L\lambda[t]\left(M[t]-m[t]\right), \qquad \forall\, t\ge 0. \qquad (13)\end{aligned}$$

The convexity of Y is crucial in establishing (13). Up to the disagreement adjustment term (i.e., $L\lambda[t]\left(M[t]-m[t]\right)$ in (13)), the above iterative relation (13) is analogous to the basic evolution in the standard centralized gradient-based method [4]. Although the basic evolution relation has been obtained for failure-free distributed algorithms [9, 19, 31, 15], since our effective objective function is time-dependent ($p_t^j(\cdot)$ in (11)), their analysis does not apply directly to our problem.

It follows from Lemma 4 that

$$\lim_{t\to\infty} L\lambda[t]\left(M[t]-m[t]\right)=0.$$

Thus, for sufficiently large t, (13) *roughly* equals

$$Dist\left(z[t+1],Y\right)\le\; \max\{\lambda[t]L,\; Dist\left(z[t],Y\right)\}.$$

Using the fact that $\lim_{t\to\infty}\lambda[t]=0$, and the "almost supermartingale" convergence theorem [21], Lemma 5 below can be proved.

LEMMA 5. *The sequence $\{Dist\left(z[t],Y\right)\}_{t=0}^{\infty}$ converges.*

Furthermore, we can show that the distance becomes 0 asymptotically, as stated in Lemma 6.

LEMMA 6. *The sequence $\{Dist\left(z[t],Y\right)\}_{t=0}^{\infty}$ converges to 0, i.e., $\lim_{t\to\infty} Dist\left(z[t],Y\right)=0$.*

Proof sketch for Lemma 6: Suppose $\lim_{t\to\infty} Dist\left(z[t],Y\right)=c$ with $c>0$. Then the sequence $\{z[t]\}_{t=0}^{\infty}$ will either accumulate to $(\max Y)+c$ or $(\min Y)-c$. *Roughly* speaking, there exists t_0, and $\rho>0$ such that

$$\begin{aligned}Dist\left(z[t+1],Y\right)&\le\; Dist\left(z[t],Y\right)-\lambda[t]\rho\\&\le\; Dist\left(z[t_0],Y\right)-\rho\sum_{\tau=t_0}^{t}\lambda[\tau]. \qquad (14)\end{aligned}$$

Since $\sum_{t=1}^{\infty}\lambda[t]=\infty$, taking limit on both sides of (14), we get $c\le-\infty$, proving a contradiction.

Lemma 6 is then used to establish that distance of $x_j[t]$ from Y converges to 0 as well.

6. CONSTRAINED OPTIMIZATION

We so far focussed on the *unconstrained* version of the optimization problem in (2). However, we can also generalize our results to the *constrained* version of problem (2) [25]. In particular, let $\mathcal{X}\subseteq\mathbb{R}$ such that $\mathcal{X}\neq$, and $\mathcal{X}$ is convex and closed. Then constrained version of (2) is stated below in (15). Observe that the output is now constrained to be in set $\mathcal{X}$.

$$\begin{aligned}&\text{output } x_o \text{ such that} \qquad (15)\\&\quad\text{there }\textbf{exists}\text{ weight vector }\boldsymbol{\alpha}\text{ for which}\\&\quad x_o\;\in\;\underset{x\in\mathcal{X}}{\operatorname{argmin}}\;\frac{1}{|\mathcal{N}|}\sum_{i\in\mathcal{N}}\alpha_i h_i(x),\\&\quad\sum_{i\in\mathcal{N}}\alpha_i=1, \quad\text{and}\quad \forall i,\;\alpha_i\ge 0.\end{aligned}$$

The algorithm SBG can be adapted to solve (15) with a simple modification of the state update in (3), by projecting $\widetilde{x}_j[t-1]-\lambda[t-1]\widetilde{g}_j[t-1]$ on to set $\mathcal{X}$. This projection guarantees that $x_j[t]$ is within the constraint set $\mathcal{X}$. However, compared to the original algorithm, such a projection introduces a *projection error* at each iteration. Specifically, the update of x_j can be written as follows, where $e_i[t-1]$ denotes the projection error, and $Projection_{\mathcal{X}}$ denotes projection on to $\mathcal{X}$.

$$\begin{aligned}x_j[t]&=\;Projection_{\mathcal{X}}\left(\widetilde{x}_j[t-1]-\lambda[t-1]\widetilde{g}_j[t-1]\right)\\&=\;\widetilde{x}_j[t-1]-\lambda[t-1]\widetilde{g}_j[t-1]+e_i[t-1]. \qquad (16)\end{aligned}$$

The projection error $e_i[t]$ can be shown to approach 0 as $t \to \infty$, and Theorem 2 holds true for the modified algorithm as well [25]. A complete algorithm description and analysis is presented in [25].

7. DISCUSSION AND FUTURE WORK

Compared with Byzantine failures, under the crash fault model, we can improve on the $\left(\frac{1}{2(|\mathcal{N}|-f|)}, |\mathcal{N}| - f\right)$– admissibility achieved in case of Byzantine faults. The algorithm SBG is modified in this case to perform *no trimming* at all, since the agents do not tamper with messages. For the modified algorithm, we have shown [24] that all the non-faulty agents (agents in $\mathcal{N}$) produce an output that equals an optimum of a global cost function of the form below.

$$c\left(\sum_{i\in\mathcal{N}} h_i(x) + \sum_{i\in\mathcal{F}} \alpha_i h_i(x)\right), \tag{17}$$

where $\mathcal{F}$ is the set of faulty agents (that crash at some point during the execution), $0 \leq \alpha_i \leq 1$ for each $i \in \mathcal{F}$ and c is a normalization constant such that $c\left(|\mathcal{N}| + \sum_{i\in\mathcal{F}} \alpha_i\right) = 1$. Note that in (17), all the local functions associated with non-faulty agents have equal weights. A finite-time interpretation of the above results is also of practical interest.

We have only considered synchronous systems so far. In an asynchronous system as well, when there are up to f Byzantine faults, algorithm SBG can be modified to achieve fault-tolerant optimization. For instance, algorithm SBG may be combined with the reliable broadcast algorithm in [1]. Alternatively, we can require $n > 5f$, and combine SBG with the simpler asynchronous iterative Byzantine consensus algorithm in [8]. The two approaches will achieve a trade-off between communication cost and optimization performance.

Open Problems

Incomplete networks.

In this paper, we assumed that the underlying communication network is a completely connected. We have also explored SBG-like algorithms [25] for incomplete networks. However, our present approach is not believed to be optimal in general in incomplete network topologies. In particular, as seen previously, algorithm SBG achieves optimal fault tolerance, while also ensuring weights (α_i's) are bounded below by an adequately large constant (particularly, $\frac{1}{2(|\mathcal{N}|-f)}$). Obtaining equally strong results for incomplete networks remains an open problem.

Vector arguments.

Algorithm SBG assumes that the domain for the argument of the cost functions is $\mathbb{R}$ (or, in case of *constrained* optimization in (15) with $\mathcal{X} = \mathbb{R}$). In general, we would like to solve problem (2) for vector (i.e., multidimensional) arguments in $\mathbb{R}^k$ for $k \geq 2$ as well. In recent work, the problem of Byzantine *vector* consensus has been solved [17, 35]. However, a solution for Byzantine vector consensus by itself is *not* adequate to be able to solve the optimization problem of interest here. The difficulty lies in the geometry of the set of optima, when the argument is a higher dimensional vector. In particular, unlike the one-dimensional case where set Y defined in (5) is convex, it is not necessarily convex when the argument is higher dimensional.

Non-smooth cost functions.

In our work, we assumed continuously differentiable cost functions. In general, the cost functions may be non-smooth, and the optimization algorithm would need to use *subgradients* instead of *gradients*. For the *failure-free* case, distributed subgradient optimization algorithms indeed exist [9, 19], however, design and analysis of fault-tolerant optimization algorithms for non-smooth cost functions remain open.

8. ACKNOWLEDGMENTS

The authors would like to thank the reviewers for insightful comments and suggestions.

9. REFERENCES

[1] I. Abraham, Y. Amit, and D. Dolev. Optimal resilience asynchronous approximate agreement. In *Principles of Distributed Systems*, pages 229–239. Springer, 2005.

[2] D. P. Bertsekas. *Convex Optimization Algorithms.* Athena Scientific, 2015.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, Jan. 2011.

[4] S. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[5] S. Chaudhuri. More choices allow more faults: Set consensus problems in totally asynchronous systems. *Information and Computation*, 105:132–158, 1992.

[6] J. Chen and A. Sayed. Diffusion adaptation strategies for distributed optimization and learning over networks. *Signal Processing, IEEE Transactions on*, 60(8):4289–4305, August 2012.

[7] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl. Reaching approximate agreement in the presence of faults. *J. ACM*, 33(3):499–516, May 1986.

[8] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl. Reaching approximate agreement in the presence of faults. *Journal of the ACM (JACM)*, 33(3):499–516, 1986.

[9] J. Duchi, A. Agarwal, and M. Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *Automatic Control, IEEE Transactions on*, 57(3):592–606, March 2012.

[10] A. D. Fekete. Asymptotically optimal algorithms for approximate agreement. *Distributed Computing*, 4(1):9–29, 1990.

[11] R. Friedman, A. Mostefaoui, S. Rajsbaum, and M. Raynal. Asynchronous agreement and its relation with error-correcting codes. *Computers, IEEE Transactions on*, 56(7):865–875, 2007.

[12] M. Herlihy, S. Rajsbaum, M. Raynal, and J. Stainer. Computing in the presence of concurrent solo executions. In *LATIN 2014: Theoretical Informatics*, pages 214–225. Springer Berlin Heidelberg, 2014.

[13] B. Johansson. On distributed optimization in networked systems. 2008.

[14] H. J. LeBlanc, H. Zhang, S. Sundaram, and X. Koutsoukos. Consensus of multi-agent networks in the presence of adversaries using only local

information. In *Proceedings of the 1st International Conference on High Confidence Networked Systems*, HiCoNS '12, pages 1–10, New York, NY, USA, 2012. ACM.

[15] I. Lobel and A. Ozdaglar. Distributed subgradient methods for convex optimization over random networks. *Automatic Control, IEEE Transactions on*, 56(6):1291–1306, June 2011.

[16] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[17] H. Mendes and M. Herlihy. Multidimensional approximate agreement in byzantine asynchronous systems. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 391–400, New York, NY, USA, 2013. ACM.

[18] A. Mostefaoui, S. Rajsbaum, and M. Raynal. Conditions on input vectors for consensus solvability in asynchronous distributed systems. *Journal of the ACM (JACM)*, 50(6):922–954, 2003.

[19] A. Nedic and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *Automatic Control, IEEE Transactions on*, 54(1):48–61, Jan 2009.

[20] Y. Nesterov. *Introductory lectures on convex optimization*, volume 87. Springer Science & Business Media, 2004.

[21] H. Robbins and D. Siegmund. A convergence theorem for non negative almost supermartingales and some applications. In T. Lai and D. Siegmund, editors, *Herbert Robbins Selected Papers*, pages 111–135. Springer New York, 1985.

[22] L. Su and N. H. Vaidya. Byzantine multi-agent optimization: Part I. *arXiv preprint arXiv:1506.04681*, 2015.

[23] L. Su and N. H. Vaidya. Byzantine multi-agent optimization: Part II. *CoRR*, abs/1507.01845, 2015.

[24] L. Su and N. H. Vaidya. Byzantine multi-agent optimization: Part III. *Technical Report*, 2015.

[25] L. Su and N. H. Vaidya. Fault-tolerant distributed optimization (Part IV): Constrained optimization with arbitrary directed networks. *arXiv preprint arXiv:1511.01821*, 2015.

[26] L. Su and N. H. Vaidya. Multi-agent optimization in the presence of byzantine adversaries: Fundamental limits. In *Proceedings of IEEE American Control Conference (ACC)*, July, 2016.

[27] L. Su and N. H. Vaidya. Fault-tolerant multi-agent optimization: Optimal distributed algorithms (full version). In *Technical Report*, May, 2016.

[28] S. Sundaram and B. Gharesifard. Consensus-based distributed optimization with malicious nodes. In *Proceedings of the 53rd Annual Allerton Conference on Communication, Control and Computing*. IEEE, 2015.

[29] L. Tseng and N. H. Vaidya. Iterative approximate byzantine consensus under a generalized fault model. In *Distributed Computing and Networking*, pages 72–86. Springer, 2013.

[30] L. Tseng and N. H. Vaidya. Iterative approximate consensus in the presence of byzantine link failures. In *Networked Systems*, pages 84–98. Springer International Publishing, 2014.

[31] K. I. Tsianos, S. Lawlor, and M. G. Rabbat. Push-sum distributed dual averaging for convex optimization. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 5453–5458, Dec 2012.

[32] J. Tsitsiklis, D. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *Automatic Control, IEEE Transactions on*, 31(9):803–812, Sep 1986.

[33] J. N. Tsitsiklis. Problems in decentralized decision making and computation. Technical report, DTIC Document, 1984.

[34] N. H. Vaidya. Iterative byzantine vector consensus in incomplete graphs. In *Distributed Computing and Networking*, pages 14–28. Springer, 2014.

[35] N. H. Vaidya and V. K. Garg. Byzantine vector consensus in complete graphs. In *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, pages 65–73. ACM, 2013.

[36] N. H. Vaidya, L. Tseng, and G. Liang. Iterative approximate byzantine consensus in arbitrary directed graphs. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, pages 365–374. ACM, 2012.

APPENDIX

A. PROOF OF LEMMA 1

Proof that set Y is convex

We first prove that set Y is convex. Let $x_1, x_2 \in Y$ such that $x_1 \neq x_2$. By definition of Y, there exist valid functions $p_1(x) = \sum_{i \in \mathcal{N}} \alpha_i h_i(x) \in \mathcal{C}$ and $p_2(x) = \sum_{i \in \mathcal{N}} \beta_i h_i(x) \in \mathcal{C}$ such that $x_1 \in \text{argmin } p_1(x)$ and $x_2 \in \text{argmin } p_2(x)$, respectively. In addition, let $p(x) = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} h_i(x)$. By definition of valid function in (4), it holds that $p(x) \in \mathcal{C}$. Note that it is possible that $p_1(\cdot) = p_2(\cdot)$, and that $p_i(\cdot) = p(\cdot)$ for $i = 1$ or $i = 2$.

Given $0 \leq \alpha \leq 1$, let $x_\alpha = \alpha x_1 + (1-\alpha)x_2$. We consider two cases:

(i) $x_\alpha \in \text{argmin } p_1(x) \cup \text{argmin } p_2(x) \cup \text{argmin } p(x)$, and

(ii) $x_\alpha \notin \text{argmin } p_1(x) \cup \text{argmin } p_2(x) \cup \text{argmin } p(x)$.

Case (i): Suppose $x_\alpha \in \text{argmin } p_1(x) \cup \text{argmin } p_2(x) \cup \text{argmin } p(x)$.

When $x_\alpha \in \text{argmin } p_1(x) \cup \text{argmin } p_2(x) \cup \text{argmin } p(x)$, by definition of Y, we have

$$x_\alpha \in \text{argmin } p_1(x) \cup \text{argmin } p_2(x) \cup \text{argmin } p(x) \subseteq Y.$$

Thus, $x_\alpha \in Y$.

Case (ii): Suppose $x_\alpha \notin \text{argmin } p_1(x) \cup \text{argmin } p_2(x) \cup \text{argmin } p(x)$.

By symmetry, without loss of generality, assume that $x_1 < x_2$. By definition of x_α and the fact that $\text{argmin}_{x \in \mathbb{R}}\, p_1(x)$ is convex, it holds that $x_1 < x_\alpha < x_2$. By assumption of case (ii), it must be that $x_\alpha > \max(\text{argmin}\, p_1(x))$ and $x_\alpha < \min(\text{argmin}\, p_2(x))$, which imply that $p_1'(x_\alpha) > 0$ and $p_2'(x_\alpha) < 0$.

There are two possibilities for $p'(x_\alpha)$ (the gradient of $p(x_\alpha)$): $p'(x_\alpha) < 0$ or $p'(x_\alpha) > 0$. Note that $p'(x_\alpha) \neq 0$ because $x_\alpha \notin \text{argmin } p(x)$.

When $p'(x_\alpha) < 0$, there exists $0 \le \zeta \le 1$ such that

$$\zeta \; p_1'(x_\alpha) + (1-\zeta) \; p'(x_\alpha) = 0.$$

By definition of $p_1(x)$ and $p(x)$, we have

$$\begin{aligned} 0 &= \zeta \; p_1'(x_\alpha) + (1-\zeta) \; p'(x_\alpha) \\ &= \zeta \; \left(\sum_{i \in \mathcal{N}} \alpha_i h_i'(x_\alpha) \right) + (1-\zeta) \left(\frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} h_i'(x_\alpha) \right) \\ &= \sum_{i \in \mathcal{N}} \left(\alpha_i \zeta + (1-\zeta) \frac{1}{|\mathcal{N}|} \right) h_i'(x_\alpha). \end{aligned}$$

Thus, x_α is an optimum of function

$$\sum_{i \in \mathcal{N}} \left(\alpha_i \zeta + (1-\zeta) \frac{1}{|\mathcal{N}|} \right) h_i(x). \tag{18}$$

Let $\mathcal{I}$ be the collection of indices defined by

$$\mathcal{I} \triangleq \{ \; i: \; i \in \mathcal{N}, \text{ and } \alpha_i \zeta + (1-\zeta) \frac{1}{|\mathcal{N}|} \quad \ge \quad \frac{1}{2(|\mathcal{N}| - f)} \; \}.$$

Next we show that $|\mathcal{I}| \ge |\mathcal{N}| - f$. Let $\mathcal{I}_1$ be the collection of indices defined by

$$\mathcal{I}_1 \triangleq \{ \; i: \; i \in \mathcal{N}, \text{ and } \alpha_i \ge \frac{1}{2(|\mathcal{N}| - f)} \; \}.$$

Since $p_1(x) \in \mathcal{C}$, then $|\mathcal{I}_1| \ge |\mathcal{N}| - f$. In addition, since $n > 3f$ and $|\mathcal{N}| = n - |\mathcal{F}| > 2f$, we have $|\mathcal{N}| < 2(|\mathcal{N}| - f)$. The following holds for each $j \in \mathcal{I}_1$.

$$\begin{aligned} \alpha_j \zeta + (1-\zeta) \frac{1}{|\mathcal{N}|} &\ge \zeta \frac{1}{2(|\mathcal{N}| - f)} + (1-\zeta) \frac{1}{|\mathcal{N}|} \\ &> \zeta \frac{1}{2(|\mathcal{N}| - f)} + (1-\zeta) \frac{1}{2(|\mathcal{N}| - f)} \\ &= \frac{1}{2(|\mathcal{N}| - f)}, \end{aligned}$$

i.e., $j \in \mathcal{I}$. Thus, $\mathcal{I}_1 \subseteq \mathcal{I}$.

Since $|\mathcal{I}_1| \ge |\mathcal{N}| - f$, we have $|\mathcal{I}| \ge |\mathcal{N}| - f$. So function (18) is a valid function in $\mathcal{C}$. Thus, $x_\alpha \in Y$.

Similarly, we can show that the above result holds when $p'(x_\alpha) > 0$.

Therefore, set Y is convex.

Proof that Y is closed

Auxiliary proposition

For each $x \in \mathbb{R}$, let $h'_{i_1(x)}(x), \cdots, h'_{i_{|\mathcal{N}|}(x)}(x)$ be a *non-increasing* order of $h_j'(x)$, for $j \in \mathcal{N}$, i.e.,

$$h'_{i_1(x)}(x) \ge \cdots \ge h'_{i_{|\mathcal{N}|}(x)}(x).$$

Note that associated with the gradient order, there is a corresponding list of non-faulty agents

$$\{i_1(x), i_2(x), \ldots, i_{|\mathcal{N}|}(x)\},$$

in which the relative ranks of non-faulty agents vary with x.

Define $r(x)$ as follows: For each $x \in \mathbb{R}$,

$$\begin{aligned} r(x) \triangleq \; &\left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) h'_{i_1(x)}(x) \\ &+ \frac{1}{2(|\mathcal{N}| - f)} \sum_{j=2}^{|\mathcal{N}| - f} h'_{i_j(x)}(x). \end{aligned} \tag{19}$$

At each $x \in \mathbb{R}$, function $r(x)$ is the largest gradient value among all the valid functions in $\mathcal{C}$.

We now prove Proposition 2 that is useful in proving set Y is closed.

PROPOSITION 2. *$r(\cdot)$ is continuous and non-decreasing.*

PROOF. Recall that $\{i_1(x), i_2(x), \ldots, i_{|\mathcal{N}|}(x)\}$ is the list of non-faulty agents that corresponds to the non-increasing gradient order $h'_{i_1(x)}(x), \cdots, h'_{i_{|\mathcal{N}|}(x)}(x)$. By definition, function

$$\left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) h_{i_1(x)}(\cdot) + \frac{1}{2(|\mathcal{N}| - f)} \sum_{j=2}^{|\mathcal{N}| - f} h_{i_j(x)}(\cdot)$$

is contained in $\mathcal{C}$. Since at each $x \in \mathbb{R}$, $r(x)$ is the largest gradient among gradients of all valid functions in $\mathcal{C}$, for any y (which may equal x) we have

$$\begin{aligned} r(y) \ge &\left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) h'_{i_1(x)}(y) \\ &+ \frac{1}{2(|\mathcal{N}| - f)} \sum_{j=2}^{|\mathcal{N}| - f} h'_{i_j(x)}(y). \end{aligned} \tag{20}$$

Now, suppose $y \ge x \in \mathbb{R}$. Since $h_i'(\cdot)$ is non-decreasing, we have

$$\begin{aligned} r(y) &\ge \left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) h'_{i_1(x)}(y) \\ &\quad + \frac{1}{2(|\mathcal{N}| - f)} \sum_{j=2}^{|\mathcal{N}| - f} h'_{i_j(x)}(y) \quad \text{by (20)} \\ &\ge \left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) h'_{i_1(x)}(x) \\ &\quad + \frac{1}{2(|\mathcal{N}| - f)} \sum_{j=2}^{|\mathcal{N}| - f} h'_{i_j(x)}(x) \\ &= r(x) \quad \text{by (19)} \end{aligned}$$

Thus, function $r(\cdot)$ is non-decreasing.

Next we show that function $r(\cdot)$ is continuous.

For each $i \in \mathcal{V}$, since $h_i(\cdot)$ is continuously differentiable, it follows that $h_i'(\cdot)$ is continuous. That is, for each $i \in \mathcal{V}$ and $\forall \, \epsilon > 0$, $\exists \, \delta > 0$ such that

$$|x - c| < \delta \implies \left| h_i'(x) - h_i'(c) \right| \le \epsilon. \tag{21}$$

Assume $c \le x < c + \delta$. Then

$$\begin{aligned}
|r(x) - r(c)| &= r(x) - r(c) \quad \text{by monotonicity of } r(\cdot)\\
&\le \left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) h'_{i_1(x)}(x)\\
&\quad + \frac{1}{2(|\mathcal{N}| - f)} \sum_{j=2}^{|\mathcal{N}|-f} h'_{i_j(x)}(x)\\
&\quad - \left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) h'_{i_1(x)}(c)\\
&\quad - \frac{1}{2(|\mathcal{N}| - f)} \sum_{j=2}^{|\mathcal{N}|-f} h'_{i_j(x)}(c) \quad \text{by (20)}\\
&\le \left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) \left(h'_{i_1(x)}(x) - h'_{i_1(x)}(c)\right)\\
&\quad + \frac{1}{2(|\mathcal{N}| - f)} \sum_{j=2}^{|\mathcal{N}|-f} \left(h'_{i_j(x)}(x) - h'_{i_1(x)}(c)\right)\\
&\le \left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) \epsilon + \frac{1}{2(|\mathcal{N}| - f)} \sum_{j=2}^{|\mathcal{N}|-f} \epsilon \quad \text{by (21)}\\
&= \left(\left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) + \frac{1}{2(|\mathcal{N}| - f)} \cdot (|\mathcal{N}| - f - 1)\right) \epsilon\\
&= \epsilon. \qquad (22)
\end{aligned}$$

Similarly, we can show that when $c - \delta < x \le c$, $|r(x) - r(c)| < \epsilon$.

Thus, function $r(\cdot)$ is continuous.

The proof of Proposition 2 is complete. □

Proof that Y is closed

With the auxiliary function $r(\cdot)$ at hand, we can show the closedness of set Y as follows.

Recall that Y is convex. To show Y is closed, it is enough to show that Y is bounded and both $\min Y$ and $\max Y$ exist.

It can be easily seen that $r(x)$ is negative for "sufficiently" small x, and positive for "sufficiently" large x. By Proposition 2, we know that function $r(x)$ is non-decreasing and continuous. Thus, there exists $x_0 \in \mathbb{R}$ such that

$$\begin{aligned}
0 \;=\; r(x_0) &= \left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) h'_{i_1(x_0)}(x_0)\\
&\quad + \frac{1}{2(|\mathcal{N}| - f)} \sum_{j=2}^{|\mathcal{N}|-f} h'_{i_j(x_0)}(x_0).
\end{aligned}$$

Let

$$\begin{aligned}
q(x) &= \left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) h_{i_1(x_0)}(x)\\
&\quad + \frac{1}{2(|\mathcal{N}| - f)} \sum_{j=2}^{|\mathcal{N}|-f} h_{i_j(x_0)}(x). \qquad (23)
\end{aligned}$$

Thus, $q'(x_0) = r(x_0) = 0$.

By construction, $q(x) \in \mathcal{C}$ is a valid function. Note that due to the possibility of existence of ties in top $|\mathcal{N}| - f$ rankings of the order $h'_{i_1(x)}(x), \cdots, h'_{i_{|\mathcal{N}|}(x)}(x)$, for a given x_0, there may be multiple orders over $h'_i(x_0), \forall i \in \mathcal{N}$ of the top $|\mathcal{N}| - f$ elements. Let $\mathcal{O}$ be the collection of all such orders. Note that there is an one-to-one correspondence of an order and a valid function defined in (23). Let

$$a = \min_{o \in \mathcal{O}} \min\left(\text{argmin } q_o(x)\right),$$

which is well-defined since argmin $q_o(x)$ is compact, and $|\mathcal{O}|$ is finite.

By definition $a \in Y$. Next we show that $a = \min Y$.

Suppose, on the contrary that, there exists $\tilde{a} < a$ such that $\tilde{a} \in Y$. Since $\tilde{a} \in Y$, there exists $\tilde{q}(x) = \sum_{i \in \mathcal{N}} \alpha_i h_i(x) \in \mathcal{C}$ such that $\tilde{a} \in \text{argmin } \tilde{q}(x)$. That is,

$$\tilde{q}'(\tilde{a}) = 0. \qquad (24)$$

We have

$$\begin{aligned}
0 = q'(\tilde{a}) &= \sum_{i \in \mathcal{N}} \alpha_i h'_i(\tilde{a})\\
&\le \left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) h'_{i_1(\tilde{a})}(\tilde{a})\\
&\quad + \frac{1}{2(|\mathcal{N}| - f)} \sum_{j=2}^{|\mathcal{N}|-f} h'_{i_j(\tilde{a})}(\tilde{a})\\
&= r(\tilde{a}).
\end{aligned}$$

From the definition of a and the assumption that $\tilde{a} < a$, we get $\tilde{a} < x_0$. Then, by monotonicity of $r(\cdot)$, we have

$$r(\tilde{a}) \le r(x_0).$$

Thus, $r(\tilde{a}) = 0 = r(x_0)$. In addition, we have

$$\begin{aligned}
0 = r(\tilde{a}) &= \left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) h'_{i_1(\tilde{a})}(\tilde{a})\\
&\quad + \frac{1}{2(|\mathcal{N}| - f)} \sum_{j=2}^{|\mathcal{N}|-f} h'_{i_j(\tilde{a})}(\tilde{a}) \quad \text{by definition of } r(\tilde{a})\\
&\le \left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) h'_{i_1(\tilde{a})}(x_0)\\
&\quad + \frac{1}{2(|\mathcal{N}| - f)} \sum_{j=2}^{|\mathcal{N}|-f} h'_{i_j(\tilde{a})}(x_0) \quad \text{by monotonicity of } h'_i(\cdot)\\
&\le \left(1 - \frac{|\mathcal{N}| - f - 1}{2(|\mathcal{N}| - f)}\right) h'_{i_1(x_0)}(x_0)\\
&\quad + \frac{1}{2(|\mathcal{N}| - f)} \sum_{j=2}^{|\mathcal{N}|-f} h'_{i_j(x_0)}(x_0), \quad \text{by (20)}
\end{aligned}$$

which implies that $i_1(\tilde{a}), \cdots, i_{|\mathcal{N}|-f}(\tilde{a})$ is an order in $\mathcal{O}$. Thus, it can be seen that $\tilde{a} \ge a = \min_{o \in \mathcal{O}} \min\left(\text{argmin } q_o(x)\right)$, contradicting the assumption that $\tilde{a} < a$.

Therefore, $a = \min Y$, i.e., $\min Y$ exists. Similarly, we can show that $\max Y$ also exists.

Therefore, set Y is closed.