Detection and Isolation of Adversaries in Decentralized Optimization for Non-Strongly Convex Objectives [⋆]

Nikhil Ravi [*] **Anna Scaglione** [*]

[*] *School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85281 USA (e-mail: {Nikhil.Ravi,Anna.Scaglione}@asu.edu).*

Abstract: Decentralized optimization has found a significant utility in recent years, as a promising technique to overcome the curse of dimensionality when dealing with large-scale inference and decision problems in big data. While these algorithms are resilient to node and link failures, they however, are not inherently Byzantine fault-tolerant towards insider data injection attacks. This paper proposes a decentralized robust subgradient push (RSGP) algorithm for detection and isolation of malicious nodes in the network for optimization non-strongly convex objectives. In the attack considered in this work, the malicious nodes follow the algorithmic protocols, but can alter their local functions arbitrarily. However, we show that in sufficiently structured problems, the method proposed is effective in revealing their presence. The algorithm isolates detected nodes from the regular nodes, thereby mitigating the ill-effects of malicious nodes. We also provide performance measures for the proposed method.

Keywords: decentralized optimization, multi-agent systems, byzantine fault-tolerance, adversarial optimization, gradient-based metric

1. INTRODUCTION

A number of machine learning and big data problems can be generalized to take the following form:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} F(\boldsymbol{x}) = \min_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{x}), \tag{1}$$

where $F(\cdot) : \mathbb{R}^d \to \mathbb{R}$ is the global objective which the nodes are trying to collectively minimize. However, the nodes only have access to their own private cost functions, $f_i(\cdot) : \mathbb{R}^d \to \mathbb{R}$. There is a large class of decentralized peer-to-peer algorithms for solving such *decomposable* optimization problems via peer-to-peer consensus protocols, see Tsitsiklis et al. (1986); Nedić and Ozdaglar (2009) for early contributions and see Boyd et al. (2011); Nedić et al. (2018) for extensive literature surveys.

While these algorithms are resilient to node or edge failures, they are not however inherently resilient to insider-based data injection attacks (see e.g. Gentz et al. (2015); Blanchard et al. (2017); Gentz et al. (2016); Wu et al. (2018); Sundaram and Gharesifard (2018)). This has spurred significant interest on robust decentralized optimization algorithms. In literature, broadly speaking, there are two schools of solutions. In the first set of works, authors propose methods to regularize the objective by relaxing the hard consensus constraints, typically by using TV norm regularization scheme, see Ben-Ameur et al. (2016); Koppel et al. (2017); Xu et al. (2018). Even if the adversarial environment can be studied similarly to the non-adversarial case, such approaches do not admit the same solutions as the original problem even when the attackers are not present at all. Also, their performance is highly dependent on the attack scenario, the choice of the regularizing function, and the regularizing parameter.

These issues are not shared by the second set of works, where each node builds a score about their neighbors' updates and uses this score to detect their potential malicious intent, see Vuković and Dán (2014); Su and Vaidya (2015a,b); Su and Shahrampour (2018); Sundaram and Gharesifard (2018); Ravi et al. (2019). These scores can then be used by regular nodes to dynamically sever ties with suspicious peers and continue to run the algorithm, see Ravi et al. (2019). As long as the network remains strongly connected and set of the isolated nodes includes all the malicious nodes, the network converges to optimum solution of the objective that only includes the nodes that are not isolated. This is possible for appropriate peer-to-peer optimization algorithms because they can restore normal operations by isolating the attackers while the algorithm is running.

The literature considers a number of attack models that differ based on the structure of the data injected by the malicious nodes. This paper focuses on attacks that involve coordinated malicious agents that follow algorithmic protocols while manipulating their private cost functions arbitrarily. We consider a network of nodes that are collectively solving problem (1) by implementing the Push-Sum

[⋆] This research was supported in part by Cybersecurity for Energy Delivery Systems program, of the U.S. Department of Energy, under contract DOE0000780 and the National Science Foundation CCF-BSF 1714672 grant. Any opinions, and findings expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

algorithm proposed in Kempe et al. (2003). Concurrently, the regular nodes maintain a *maliciousness* score about their neighbors as a function of the local neighborhood data. The idea is to sever ties with those deemed to be malicious and then continuing the updates according to Push-Sum. This approach leverages the self-healing property of the gradient-based Push-Sum algorithm where nodes dynamically sever ties with those neighbors that are more likely to be malicious. The paper is organized as follows. In Section 2, the system model and the attack characteristics are described. In Section 3, we introduce the Robust Subgradient Push algorithm and its consequences. In Section 4, we present some simulation results that demonstrate the effectiveness of the proposed approach. We conclude in Section 5.

Notations: Boldfaced lower-case (resp. upper-case) letters denote vectors (resp. matrices) and x_i (X_{ij}) denotes the ith element of a vector $\boldsymbol{x}$ (the ijth entry of a matrix $\boldsymbol{X}$). Calligraphic letters are sets and $|\mathcal{A}|$ denotes the cardinality of a set $\mathcal{A}$; difference between two sets $\mathcal{A}$ and $\mathcal{B}$ is denoted by $\mathcal{A} \setminus \mathcal{B}$.

2. SYSTEM MODEL

The set of nodes in the system are modeled by a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges that represent the communication links between the nodes. To describe the adversarial environment, the set $\mathcal{V}$ is partitioned into two subsets, namely: the set $\mathcal{V}_r$ of regular nodes (RNs) with $N_r := |\mathcal{V}_r|$, and the set $\mathcal{V}_m$ of malicious nodes (MNs) with $N_m := |\mathcal{V}_m|$, so that $\mathcal{V} = \mathcal{V}_r \cup \mathcal{V}_m$. Similarly, the edge set $\mathcal{E}$ is also partitioned $\mathcal{E} = \mathcal{E}_r \cup \mathcal{E}_m$, where $\mathcal{E}_r$ is the set of links emanating from RNs, while $\mathcal{E}_m$ is the set of links emanating from the MNs. The set $\mathcal{E}_r$ is further partitioned $\mathcal{E}_m = \mathcal{E}_{mr} \cup \mathcal{E}_{mm}$, while $\mathcal{E}_{mm}$ is the set of links emanating and ending at MNs. The goal of the RNs is to identify and sever all the edges belonging to $\mathcal{E}_{mr}$. Let $\mathcal{N}_i^-$ be the set of all nodes that can send information to node i and $\mathcal{N}_i^+$ be the set of all nodes that can receive information from node i.

Problem Statement: In the presence of adversaries in a network, the benchmark for th performance is the analogous problem as in (1), where the graph is replaced with the graph formed by the regular nodes only, i.e.,

$$\underset{\{\boldsymbol{x}_i, i \in \mathcal{V}_r\}}{\text{minimize}} \frac{1}{|\mathcal{V}_r|} \sum_{i \in \mathcal{V}_r} f_i(\boldsymbol{x}_i) \text{ s.t. } \boldsymbol{x}_i = \boldsymbol{x}_j \; \forall \; ij \in \mathcal{E}_{rr}, \tag{2}$$

where $f_i : \mathbb{R}^d \to \mathbb{R}$ is the private cost/loss function only available at node i and the constraint enforces consensus in the network's neighborhoods. Decentralized optimization algorithms designed for arbitrary network topologies require doubly-stochastic weight matrices. Techniques such as Subgradient-Push (SGP) relax this requirement Nedić et al. (2018). Further improvements in the convergence rates algorithms for directed graphs are presented in DEXTRA (Xi and Khan (2017)) and ADD-OPT/Push-DIGing (Xi et al. (2018)). However, these algorithms require each node to know its out-degrees (the number of its out-neighbors) which is not a realistic assumption in general, and is even more problematic in an adversarial environment.

Before moving on to describe the characterization of malicious nodes, we introduce a few assumptions.

Assumption 1. (Strong Connectivity) The sequence of graphs, $\{\mathcal{G}(t)\}$, induced by the dynamic in- and out-neighborhoods over time is strongly connected.

Assumption 2. (Convexity) Each of the functions $f_i, \forall i \in \mathcal{V}$, is convex over the $\mathcal{R}^d$.

Assumption 3. (Bounded Subgradients) All the subgradients of each of the functions $f_i, \forall i \in \mathcal{V}$, are bounded. That is, $\exists \, l_i < \infty$ such that $\|\boldsymbol{g}_i\| \leq l_i, \forall i \in \mathcal{V}$, for all subgradients $\boldsymbol{g}_i$ of $f_i(\boldsymbol{x})$ and for all $\boldsymbol{x} \in \mathbb{R}^d$.

Assumption 4. (Malicious nodes) Let us suppose that the sequence of graphs $\{\mathcal{G}(t)\}$ is $(2\kappa + 1)$-strongly connected. Then there exists at most κ adversaries in the whole network, i.e., $|\mathcal{V}_m| \leq \kappa$. This also implies that there exists at most κ adversaries in a node's in-neighborhood, i.e., $|\mathcal{N}_i^- \cup \mathcal{V}_m| \leq \kappa, \forall i \in \mathcal{V}$.

If assumptions 1-3 hold and under no interference from malicious nodes, the convergence of the original Gradient-Push was proven in Nedić and Olshevsky (2015), i.e.,

$$\boldsymbol{x}_i^\infty \triangleq \lim_{t \to \infty} \boldsymbol{x}_i(t) = \boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \mathbb{R}^d} F(\boldsymbol{x}), \;\; \forall \;\; i \in \mathcal{V}$$

2.1 Attack characterization

Let $F^r(\boldsymbol{x})$ be the cost and $\boldsymbol{x}^*$ the solution of (2), i.e.:

$$F^r(\boldsymbol{x}) \triangleq \frac{1}{|\mathcal{V}_r|} \sum_{i \in \mathcal{V}_r} f_i(\boldsymbol{x}) \; , \;\; \boldsymbol{x}^* \triangleq \arg\min_{\boldsymbol{x}} \sum_{i \in \mathcal{V}_r} f_i(\boldsymbol{x}). \tag{3}$$

The performance of any algorithm that is resilient towards an attack can be measured using several metrics that capture the deviation of the limiting state from the ideal asymptotic condition in (3), for example:

- **Average solution difference**:
$$\epsilon_p \triangleq \frac{1}{|\mathcal{V}_r|} \sum_{i \in \mathcal{V}_r} \|\boldsymbol{x}_i^\infty - \boldsymbol{x}^*\|_p \tag{4}$$
- **Average cost increase**:
$$\varrho \triangleq \frac{1}{|\mathcal{V}_r|} \sum_{i \in \mathcal{V}_r} f_i(\boldsymbol{x}_i^\infty) - F^r(\boldsymbol{x}^*) \tag{5}$$
- **Average deviation from consensus**:
$$\gamma \triangleq \frac{1}{|\mathcal{V}_r|} \sum_{i \in \mathcal{V}_r} \|\boldsymbol{x}_i^\infty - \overline{\boldsymbol{x}}^\infty\| \, , \;\; \overline{\boldsymbol{x}}^\infty \triangleq \frac{1}{|\mathcal{V}_r|} \sum_{\ell \in \mathcal{V}_r} \boldsymbol{x}_\ell^\infty \tag{6}$$

It is a well established that the presence of even a single malicious node will lead the convergence of the algorithm astray (see e.g. Figure 1). The type of attack that is considered in our paper, represents the case where the regular nodes' functions $f_i(\boldsymbol{x})$, $i \in \mathcal{V}_r$ have common statistical characteristics. The attacker follows the algorithm as well, however, its intent is to alter the solution, which requires altering the statistics of the function and its gradient. For example, this situation arises when the distributed optimization objective is to solve a regression problem where, in normal conditions, the sensors cooperatively are trying to estimate a vector $\boldsymbol{x}_o$ based on a noisy observation model:

$$s_i = h_i(\boldsymbol{x}_o) + w_i \;\; \in \; \mathbb{R} \tag{7}$$

for all $i \in \mathcal{V}$ where w_i are independent identically distributed noise samples. The regression can be formulated

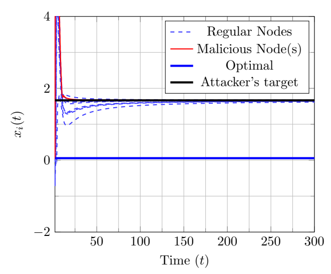

Fig. 1. Consensus without attacker detection with adversaries following algorithmic protocols but modifying their local cost functions dynamically to reach their intended target (in black). Here, the regular nodes in blue converge to the attacker's (in red) target. In the example a set of nodes $(1-10)$ are trying to solve problem 1 using the standard Sub-gradient push (SGP) in a static network. However, a malicious node drives the network towards its target, x^a.

as the minimization of the sum of $f_i(\boldsymbol{x}) = \|s_i - h_i(\boldsymbol{x})\|^2$, which amounts to seeking the maximum $\boldsymbol{x}$. To be concrete, we will focus on Gaussian independent identically distributed (i.i.d.) noise case $w_i \sim \mathcal{N}(0, \sigma^2)$ in which case:

$$f_i(\boldsymbol{x}) = \|s_i - h_i(\boldsymbol{x})\|^2 \tag{8}$$

and the gradient [1] is:

$$\boldsymbol{g}_i(\boldsymbol{x}) = 2\nabla h_i(\boldsymbol{x})(s_i - h_i(\boldsymbol{x})) \tag{9}$$

The attack can be caused by a spoofing attack to a set of sensors:

$$s_m^a = s_m + \delta s_m \quad m \in \ \mathcal{V}_m \tag{10}$$

Under this attack, the gradient of the functions $f_m^a(\boldsymbol{x}) = \|s_m^a - h_m(\boldsymbol{x})\|^2$ is, therefore perturbed:

$$\boldsymbol{g}_m(\boldsymbol{x}) = 2\nabla h_m(\boldsymbol{x})(s_m^a - h_m(\boldsymbol{x})) = \boldsymbol{g}_m(\boldsymbol{x}) + \delta \boldsymbol{g}_m(\boldsymbol{x}) \tag{11}$$

Under such an attack, a strongly connected network will converge to consensus on the following solution:

$$\boldsymbol{x}^a = \arg\min_{\boldsymbol{x}} \left(\frac{|\mathcal{V}_r|}{|\mathcal{V}|} F^r(\boldsymbol{x}) + \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}_m} f_i^a(\boldsymbol{x}) \right) \tag{12}$$

with average deviation from consensus $\gamma = 0$ and average solution difference and cost deviation, respectively:

$$\epsilon_p = \|\boldsymbol{x}^a - \boldsymbol{x}^*\|_p, \quad \varrho = F^r(\boldsymbol{x}^a) - F^r(\boldsymbol{x}) \tag{13}$$

In the context of this type of attack, the percentage increase the additional average error relative to the latent parameter $\boldsymbol{x}_o$ measures the damage caused by the malicious agents as well, that is for instance:

$$\xi_p \triangleq \frac{\frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} \|\boldsymbol{x}_i^\infty - \boldsymbol{x}_o\|_p}{\|\boldsymbol{x}^* - \boldsymbol{x}_o\|_p} \tag{14}$$

where a significant damage comes from a large value of ξ_p in the presence of an attack; since when the attack is fully successful there is consensus on $\boldsymbol{x}^a$ the value of ξ_p is:

$$\xi_p = \|\boldsymbol{x}^a - \boldsymbol{x}_o\|_p / \|\boldsymbol{x}^* - \boldsymbol{x}_o\|_p \tag{15}$$

Of course, the quantity that we consider a benchmark $\|\boldsymbol{x}^* - \boldsymbol{x}_o\|_p$ itself represents a loss relative to the case where no sensor is attacked, as more observations improve the estimation error performance.

[1] Note that in this case, given that the noise terms are non bounded, the condition stated as Assumption 3 of bounded gradients can only be guaranteed with high probability. To avoid complications in the discussion we will discuss convergence as if the condition is met.

In our previous work Ravi et al. (2019) we noted that staging a sufficiently strong attack in terms of ϵ_p, requires either sufficiently large gradients for the malicious nodes, or a special choice for the target point $\boldsymbol{x}^a$. In fact, since $\boldsymbol{x}^a$ is the stationary point for the problem in (12), we have

$$\textstyle\sum_{i\in\mathcal{V}_r} \boldsymbol{g}_i(\boldsymbol{x}^a) + \sum_{i\in\mathcal{V}_m} \boldsymbol{g}_i(\boldsymbol{x}^a) = \boldsymbol{0}.$$

Let $\boldsymbol{H}_i(\boldsymbol{x})$ be the Hessian of the function $f_i(\boldsymbol{x})$ and $\boldsymbol{H}^r(\boldsymbol{x})$ the Hessian of the function $F^r(\boldsymbol{x})$; then:

$$\begin{aligned}\textstyle\sum_{i\in\mathcal{V}_m} \boldsymbol{g}_i(\boldsymbol{x}^a) &\approx -\textstyle\sum_{i\in\mathcal{V}_r} \boldsymbol{g}_i(\boldsymbol{x}^*) + \boldsymbol{H}_i(\boldsymbol{x}^*)(\boldsymbol{x}^a - \boldsymbol{x}^*) \quad (16)\\ &= -\boldsymbol{H}^r(\boldsymbol{x}^*)(\boldsymbol{x}^a - \boldsymbol{x}^*). \quad (17)\end{aligned}$$

The last equation implies that the most effective attacks can be staged when the malicious nodes can be selected so that the Hessian $\boldsymbol{H}^r(\boldsymbol{x}^*)$ is singular, compromising the convergence of the algorithm even if the malicious nodes were isolated and the network of regular nodes remains strongly connected.

This requires us to state the following assumption:

Assumption 5. The Hessian $\boldsymbol{H}^r(\boldsymbol{x}^*)$ of $F^r(\boldsymbol{x})$ is positive definite.

Proposition 1. (Ravi et al. (2019)). Under Assumption 5, the nodes achieve consensus (i.e. $\gamma = 0$), and the average solution difference is such that:

$$\left\|\textstyle\sum_{i\in\mathcal{V}_m} \boldsymbol{g}_i(\boldsymbol{x}^a)\right\|_2^2 \gtrsim \epsilon_2 \lambda_{\min}(\boldsymbol{H}^r(\boldsymbol{x}^*)). \tag{18}$$

3. ROBUST SUBGRADIENT PUSH

The basic idea of this paper is to leverage the structure of the problem, as well as the need of the malicious agents to use large gradients to influence the result, as it is made apparent by equation (18). We wish to use this fact to design a score $S_{ij}(t)$ that each node updates $\forall j \in \mathcal{N}_i$ and uses to detect and then reject updates from a specific node.

In this paper we propose a modified version of the Subgradient-Push (Nedić and Olshevsky (2015)) that is based on the original Push sum algorithm (Kempe et al. (2003)), see algorithm 1. This algorithm differs from the original Subgradient-Push in the fact that at all times $t > 0$, the out-degree of all the nodes is 2, i.e., instead of a broadcast communication model, each node chooses one of its out-neighbors uniformly at random and sends its information to that node and to itself. This eliminates the necessity of each node knowing its out-degree, the trade-off being a slower convergence rate. The algorithm converges even when the functions are non strongly convex, which is the typical scenario of interest in a network of sensors that collect a scalar measurement while the goal is to estimate a d-dimensional vector parameter $\boldsymbol{x}_o$.

We name it Robust Subgradient-Push (RSGP). In RSGP each node i maintains a set of variables, $\boldsymbol{z}_i(t), \boldsymbol{v}_i(t) \in \mathbb{R}^d$ and $y_i(t) \in \mathbb{R}$, at all times $t \geq 0$. The algorithm starts with an arbitrarily initiated $x_i(0) \in \mathbb{R}^d$, and with $y_i(0) = 1$ at each node i, and is descried in Algorithm 1. Here, $\boldsymbol{g}_i(t) = \nabla f_i(\boldsymbol{x}_i(t))$ is a subgradient of $f_i(\cdot)$ calculated at $\boldsymbol{x}_i(t)$, and $\eta_t > 0$ is the step-size that decays with time such that

$$\textstyle\sum_t \eta_t = \infty, \quad \sum_t \eta_t^2 < \infty. \tag{23}$$

Algorithm 1 Robust Subgradient-Push (RSGP)

```
1:  Initialize: $\forall j \in \mathcal{N}_i^-$
        $z_i(0) \in \mathbb{R}^d$ arbitrarily, $y_i(0) = 1$,
        $\tilde{S}_{ij}(0) = \mathbf{0}$
2:  Ensure:
        $\{\eta\}_t$ satisfies condition (23) and $\alpha < 1$
3:  for $t \geq 0$ do
4:      for $i \in \mathcal{V}$ do
5:          Let $\{z_j(t), y_j(t)\}, \forall j \in \mathcal{N}_i^-(t) \subseteq \mathcal{N}_i^-$, be the
            pairs of data sent to node $i$ at time $t$
6:          Send $\{z_i(t), y_i(t)\}$ to $i$ itself and a node $k \in \mathcal{N}_i^+$
            chosen uniformly at random; $\mathcal{N}_i^+(t) = \{i, k\}$
7:          Update $v_i, y_i, x_i$ and $z_i$ as follows:
                $v_i(t+1) = \frac{1}{2}\sum_{j\in\mathcal{N}_i^-(t)} z_j(t)$   (19)
                $y_i(t+1) = \frac{1}{2}\sum_{j\in\mathcal{N}_i^-(t)} y_j(t)$   (20)
                $x_i(t+1) = v_i(t+1)/y_i(t+1)$   (21)
                $z_i(t+1) = v_i(t+1) - \eta_{t+1} g_i(t+1)$   (22)
8:          if $i \in \mathcal{V}_r$ then
9:              for all $j \in \mathcal{N}_i^-$ do
10:                 Calculate the score $\tilde{S}_{ij}(t)$ from equation (31).
11:             Calculate $\chi_i(t)$ from equation (32).
12:             for all $j \in \mathcal{N}_i^-$ do
13:                 if $\tilde{S}_{ij}(t) > \chi(t)$ then
14:                     $\mathcal{N}_i^- \leftarrow \mathcal{N}_i^- \setminus j$ and $\mathcal{N}_i^+ \leftarrow \mathcal{N}_i^+ \setminus j$
```

3.1 Detection and Isolation via a Neighborhood Score

Crucial to RSGP is the definition of the score that shall reveal the malicious nodes. We note that each node receives its neighbors' $\boldsymbol{z}_i(t)$ and $y_i(t)$ state variables when they communicate. The variable $y_i(t)$ of node i is a function of the number of instantaneous in-neighbors i receives at any particular time instant. Importantly, it does not depend on node i's gradient, which is what the sensor spoofing attack manipulates. The iterative update of variable $\boldsymbol{z}_i(t)$, however, includes a gradient step. Thus, we define a *maliciousness* score $S_{ij}(t)$ that can be maintained by each regular node i about their neighbors $\mathcal{N}_i^-$. As the nodes approach consensus, their direction of descent is typically dominated by the malicious nodes' gradients. The intuition is that node i can use:

$$\hat{\boldsymbol{x}}_j(t) := \boldsymbol{z}_j(t)/y_j(t) \tag{24}$$

to track neighbor j's instantaneous gradient. From the bound in Equation (18), we hypothesize that the malicious nodes will appear as outliers relative to the rest of the nodes in the neighborhood, as long as they are sufficiently outnumbered. The instantaneous score we propose is:

$$S_{ij}(t) \triangleq \frac{1}{\eta_t^2}\left\| \sum_{\ell\in\mathcal{N}_i^-(t)\setminus j} (\hat{\boldsymbol{x}}_j(t) - \hat{\boldsymbol{x}}_\ell(t)) \right\|_2^2. \tag{25}$$

To gain some intuition, suppose the nodes are approaching consensus. At this point the values of $y_i(t)$ are likely to have converged as well, and:

$$\boldsymbol{x}_j(t) = \boldsymbol{v}_j(t)/y_j(t) \approx \boldsymbol{x}^a$$

In this case:

$$\begin{aligned}\hat{\boldsymbol{x}}_j(t+1) - \hat{\boldsymbol{x}}_\ell(t+1) &= \frac{\boldsymbol{v}_j(t)}{y_j(t)} - \frac{\boldsymbol{v}_\ell(t)}{y_\ell(t)} - \eta_t(\boldsymbol{g}_j(t) - \boldsymbol{g}_\ell(t)) \\ &\approx \eta_t(\boldsymbol{g}_\ell(t) - \boldsymbol{g}_j(t)), \end{aligned} \tag{26}$$

which indicates that the score is essentially comparing the disparity in the gradients and:

$$S_{ij}(t) \approx \left\| \sum_{\ell\in\mathcal{N}_i^-(t)\setminus j} (\boldsymbol{g}_\ell(t) - \boldsymbol{g}_j(t)) \right\|_2^2 \tag{27}$$

$$= \left\| \sum_{\ell\in\mathcal{N}_i^-(t)\setminus j} \boldsymbol{g}_\ell(t) - (d_i(t)-1)\boldsymbol{g}_j(t) \right\|_2^2. \tag{28}$$

Let us consider for simplicity the presence of only one malicious agent in the ith node neighborhood. In the following, we denote node i degree by $d_i(t) = |\mathcal{N}_i(t)|$. The last equation, together with the observation in Proposition 1, implies that the score for a malicious neighbor m, for t sufficiently large is approximately:

$$S_{im}(t) \approx (d_i(t)-1)^2 \|\boldsymbol{g}_m(t)\|_2^2. \tag{29}$$

On the other hand, it is not difficult to show expanding the norm squared in (28) that:

$$\begin{aligned}\frac{1}{d_i}\sum_{i\in\mathcal{N}_i(t)} S_{ij}(t) = d_i(t) \sum_{i\in\mathcal{N}_i(t)} \|\boldsymbol{g}_i(t)\|^2 + \qquad (30)\\ -\left(2 - \frac{1}{d_i(t)}\right)\left\| \sum_{i\in\mathcal{N}_i(t)} \boldsymbol{g}_i(t)\right\|^2 \lessapprox d_i(t)\|\boldsymbol{g}_m(t)\|_2^2\end{aligned}$$

which suggests that, as long as $(d_i(t)-1)^2 > d_i(t)$ (which happens as long as $d_i(t) \geq 4$) the malicious agent score will tend to dominate the average of the scores of the neighborhood. Considering the randomness that exist in the protocol iterations, the score used to detect the agents is averaged over time. In fact, RSGP uses a cumulative score is given by:

$$\tilde{S}_{ij}(t) = \sum_{\tau=1}^{t} \alpha^\tau S_{ij}(\tau) = \tilde{S}_{ij}(t-1) + \alpha^t S_{ij}(t), \tag{31}$$

where $0 < \alpha < 1$. The weighted average is designed in such a manner that older instantaneous scores are given lower weights than newer ones, for which (26) is more accurate.

Regular Nodes can then dynamically isolate those neighbors whose score cross a certain threshold, $\chi_i(t)$. Let $\tilde{\boldsymbol{S}}_i(t)$ be the vector of scores at node i. In our implementation we choose $\chi_i(t)$ as follows:

$$\chi_i(t) = \text{avg}\left(\tilde{\boldsymbol{S}}_i(t)\right) + \beta \times \text{std}\left(\tilde{\boldsymbol{S}}_i(t)\right) \tag{32}$$

where avg($\boldsymbol{a}$) and std($\boldsymbol{a}$) are sample average and sample standard deviation of vector $\boldsymbol{a}$ entries. The parameter β controls the aggressiveness of the edge severing strategy.

It is important to note that as a result of edge severing, depending on the aggressiveness of the isolation strategy, some regular nodes might also get isolated from the rest of the regular nodes. This may result as a consequence of the regular node being slow in isolating neighboring malicious node(s). In another scenario, the algorithm might give rise to a splintering in the network to multiple connected subgraphs $\overline{\mathcal{G}}_\ell$, thereby leading to polarities in the final convergence points of such subgraphs to $\boldsymbol{x}_\ell^\infty$. Ideally, the RSGP at the optimum value of β separates $\mathcal{G}$ into $\mathcal{G}_r$ made up solely of the regular nodes, and if the malicious nodes are coordinating (which they are in this paper), into $\mathcal{G}_m$ made up solely of the malicious nodes. If the malicious nodes are not cooperating, then we may see further splintering in $\mathcal{G}_m$. The various scenarios and their error plots are discussed in section 4.

4. NUMERICAL RESULTS WITH A CASE STUDY

To illustrate our approach, let us consider a parameter estimation problem where the observation model is given by $s_i = \boldsymbol{h}_i^{\mathrm{T}} \boldsymbol{x}_o + w_i$, where $\boldsymbol{h}_i \in \mathbb{R}^d$, $w_i \in \mathbb{R}$ represents i.i.d noise samples drawn from a normal distribution with mean zero and variance σ_i^2, for all $i \in \mathcal{V}$. Then the local linear least square loss function may be written as:

$$f_i(\boldsymbol{x}) = (\boldsymbol{h}_i^{\mathrm{T}} \boldsymbol{x}_i - s_i)^2.$$

The global cost function is $F(\boldsymbol{x}) = ({}^1/_{|\mathcal{V}|}) \sum_{i\in\mathcal{V}} f_i(\boldsymbol{x})$ when all the nodes are performing regularly. However, when certain nodes are attacked or are malicious, the regular nodes estimate $\boldsymbol{x}^*$ by solving the following problem

$$\min_{\boldsymbol{x}} F^r(\boldsymbol{x}) = \min_{\boldsymbol{x}} \frac{1}{|\mathcal{V}_r|} \sum_{i\in\mathcal{V}_r} (\boldsymbol{h}_i^{\mathrm{T}} \boldsymbol{x}_i - s_i)^2.$$

Let us assume without loss of generality that $\mathcal{V}_r = \{1, \ldots, N_r\}$ and define the matrix $\boldsymbol{A}_r^{\mathrm{T}} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{N_r}]$. Denoting by $\boldsymbol{A}_r^\dagger := \left(\boldsymbol{A}_r^{\mathrm{T}} \boldsymbol{A}_r\right)^{-1} \boldsymbol{A}_r^{\mathrm{T}}$, which is the pseudoinverse of matrix $\boldsymbol{A}_r$, the minimizer for this problem is known in closed-form and given by

$$\boldsymbol{x}^* = \boldsymbol{A}_r^\dagger \boldsymbol{s}_r = \boldsymbol{x}_o + \left(\boldsymbol{A}_r^{\mathrm{T}} \boldsymbol{A}_r\right)^{-1} \boldsymbol{A}_r^{\mathrm{T}} \boldsymbol{w}_r.$$

and $\boldsymbol{A}_r^{\mathrm{T}} \boldsymbol{A}_r$ is the Hessian, which under Assumption 5 is full rank. The cost is:

$$F^r(\boldsymbol{x}^*) = \|(\boldsymbol{I} - \boldsymbol{A}_r^\dagger \boldsymbol{A}_r)\boldsymbol{s}\|^2 = \|(\boldsymbol{I} - \boldsymbol{A}_r^\dagger \boldsymbol{A}_r)\boldsymbol{w}\|^2 \qquad (33)$$

Now, let $\boldsymbol{x}_i^\infty, \forall i \in \mathcal{V}_r$, be the points at which the regular nodes converge at the end of RSGP. The average solution error with respect to $\boldsymbol{x}^*$ is given by ϵ_p from equation (4). The average cost increase of RSGP is given by ϱ from equation (5). We can also define a regret in the errors as ξ_p from equation (14), which compares the error under attack and the error without an attack. We redefine the terms for ease of reading:

$$\epsilon_p \triangleq \frac{1}{N_r} \sum_{i\in\mathcal{V}_r} \|\boldsymbol{x}_i^\infty - \boldsymbol{x}^*\|_p, \quad \varrho \triangleq \frac{1}{N_r} \sum_{i\in\mathcal{V}_r} \left(f_i(\boldsymbol{x}_i^\infty) - f_i(\boldsymbol{x}^*)\right)$$

$$\gamma_p \triangleq \frac{1}{N_r} \sum_{i\in\mathcal{V}_r} \|\boldsymbol{x}_i^\infty - \overline{\boldsymbol{x}}^\infty\|_p, \quad \xi_p \triangleq \frac{\frac{1}{N_r} \sum_{i\in\mathcal{V}_r} \|\boldsymbol{x}_i^\infty - \boldsymbol{x}_o\|_p}{\|\boldsymbol{x}^* - \boldsymbol{x}_o\|_p},$$

where $\overline{\boldsymbol{x}}^\infty \triangleq \frac{1}{|\mathcal{V}_r|} \sum_{\ell\in\mathcal{V}_r} \boldsymbol{x}_\ell^\infty$ and $p = 2$.

The figures in Figure 2 show the performance of RSGP and compares it with the methods proposed in Ben-Ameur et al. (2016) and Sundaram and Gharesifard (2018). The system setup is as follows. We consider multiple realization of Erdős–Rényi networks with $N = 20$ nodes and $p = 3\log(N)/N$. Without loss of generality, the node sets $\mathcal{V}_r = \{1, \ldots, 17\}$ and $\mathcal{V}_m = \{18, 19, 20\}$ are kept the same for every montecarlo trial, but the edge set $\mathcal{E}$ is changed. The vector $\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \in \mathbb{R}^d$ $(d = 2)$ is drawn randomly at the start of each montecarlo trial for all $i \in \mathcal{V}$. The following parameters/system variables remain the same for all the montecarlo trials; $\boldsymbol{x}_o \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \in \mathbb{R}^d$, $\boldsymbol{h}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\sigma^2 I}) \in \mathbb{R}^d$, $\boldsymbol{s_i} = \boldsymbol{h}_i^{\mathrm{T}} \boldsymbol{x}_o + \boldsymbol{w}_i$, $\boldsymbol{w_i} \sim \mathcal{N}(0,1)$, $\forall i \in \mathcal{V}_r$. Note that $\boldsymbol{x}_o$ was initialized to $[0.0859, -1.4916]^{\mathrm{T}}$, and the malicious nodes $(i \in \mathcal{V}_m)$ alter their $\boldsymbol{h}_i$ as $\left((1/N_r)\mathbf{1}^{\mathrm{T}} \boldsymbol{A}^r - 5\right)$ and s_i as $\left((1/N_r)\mathbf{1}^{\mathrm{T}} \boldsymbol{s}^r + 5\right)$. One such realization is shown in figure 2a. In this network, nodes in color red are attacking the network by altering the parameters of its loss function f_i, $\forall i \in \mathcal{V}_m$. Let us suppose that the regular nodes keep a track of the metric in equation (31) over time. Each node isolates those neighbors that exceed the threshold in equation (32) with $\beta = 1.5$. Figure 2b shows the convergence of RSGP. At time $t = 401$ (indicated in magenta), all the nodes with the malicious node(s) in their neighborhood have successfully isolated the malicious agents. We then see the self-healing property of distributed consensus algorithm come into play and the regular nodes converge at the optimal consensus point in **blue**.

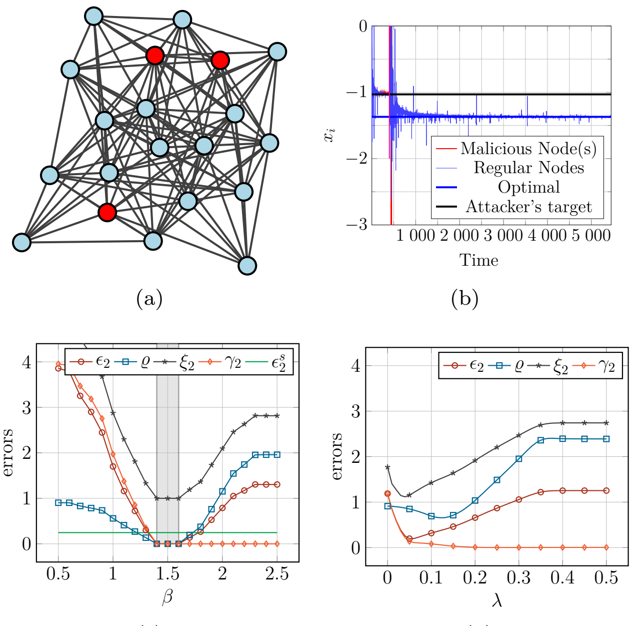

Fig. 2. (a) Erdős–Rényi network under consideration. (b) Convergence to optimal consensus point x^*. (c) Average Residual as a function of β; ϵ_2^s is the solution difference for the algorithm in Sundaram and Gharesifard (2018). (d) Average Residual as a function of λ for the method in Ben-Ameur et al. (2016).

In Figure 2c, we plot $\epsilon_2, \varrho, \xi_2$, and γ_p as functions of β, averaged over multiple montecarlo trials for realizations of $\mathcal{G}$. We see that at lower values of β, where nodes sever ties rather aggressively, the network is broken into multiple strongly connected subgraphs or into a completely disconnected network. This results in errors greater than zero. Whereas, at the intermediate values of β, where the regular nodes sever ties in a relatively passive manner, RSGP isolates almost all, if not all, the malicious nodes in the network and thus, we see a decrease in the errors. At the optimum choice for β ($\beta^* \in (1.4, 1.6)$ in this setup), the RSGP algorithm almost certainly disconnects the malicious nodes from the regular nodes and thus the errors reduce to zero. However, as β increases past this region, the nodes start to become highly conservative in their cutting, leaving almost all the malicious nodes still connected to the network. Thus, we see an increase in the errors and after a certain value of β, when no edge cutting takes place, the errors saturate. The solution difference for the method in Sundaram and Gharesifard (2018) is plotted as ϵ_2^s. In this algorithm, nodes communicate with their neighbors at all time steps, but only use those neighbors' data which are not among the extremes in the sorted set of neighborhood data points. Notice here that the malicious nodes might persist in the filtered neighborhood of regular nodes over time. Thus, this algorithm can only guarantee convergence in the convex hull of the minimizers of the regular nodes' private functions.

In Figure 2d, we plot the errors produced by the algorithm proposed in (Ben-Ameur et al., 2016, Algorithm 1), where the authors replace the consensus constraint in the problem in (1) with a TV norm regularizer to the objective, thereby penalizing nodes for not being in consensus with their neighbors. The problem is given by:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} f_i(\boldsymbol{x}_i) + \lambda \sum_{ij\in\mathcal{E}} (\boldsymbol{x}_i - \boldsymbol{x}_j), \tag{34}$$

where λ is the regularization parameter which controls the strength of magnitude of the penalty. If $\lambda = 0$, consensus among the nodes is not enforced, thereby leading the nodes to their individual local minimizers, a node i converges to the minimizer of f_i, $\forall i \in \mathcal{V}$. As λ increases past zero, the consensus constraint is enforced, indicated by γ_2 reaching zero. While this method is successful in imposing consensus among the regular nodes, it can not, unlike RSGP, reduce the ϵ_p to zero.

5. CONCLUSION

A robust decentralized optimization algorithm (RSGP) resilient to malicious Byzantine agents is proposed. This algorithm forgoes the need for the knowledge of out-degrees and works even for non-strongly convex loss functions. The algorithm dynamically isolates malicious nodes in the system thereby leading the system to convergence at the optimum consensus point. The isolation strategy is local to each node, is independent of the network topology and the attack strategy, and leverages the structure of the regular nodes in the system.

REFERENCES

Ben-Ameur, W., Bianchi, P., and Jakubowicz, J. (2016). Robust Distributed Consensus Using Total Variation. *IEEE Transactions on Automatic Control*. doi: 10.1109/TAC.2015.2471755.

Blanchard, P., Guerraoui, R., Stainer, J., and others (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, 119–129.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine learning*, 3(1), 1–122. doi:10.1561/2200000016.

Gentz, R., Wu, S.X., Wai, H.T., Scaglione, A., and Leshem, A. (2016). Data injection attacks in randomized gossiping. *IEEE Transactions on Signal and Information Processing over Networks*, PP(99), 1. doi: 10.1109/TSIPN.2016.2614898.

Gentz, R., Wai, H.T., Scaglione, A., and Leshem, A. (2015). Detection of data injection attacks in decentralized learning. In *2015 49th Asilomar Conference on Signals, Systems and Computers*, 350–354. IEEE.

Kempe, D., Dobra, A., and Gehrke, J. (2003). Gossip-Based Computation of Aggregate Information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, 482–491.

Koppel, A., Sadler, B.M., and Ribeiro, A. (2017). Proximity without consensus in online multiagent optimization. *IEEE Transactions on Signal Processing*, 65(12), 3062–3077. doi:10.1109/TSP.2017.2686368.

Nedić, A., Olshevsky, A., and Rabbat, M. (2018). Network topology and communication computation trade-offs in decentralized optimization. In *Proceedings of the IEEE*, volume 106, 953–976.

Nedić, A. and Olshevsky, A. (2015). Distributed Optimization over Time-varying Directed Graphs. *IEEE Transactions on Automatic Control*, 60(3), 601–615.

Nedić, A. and Ozdaglar, A. (2009). Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1), 48–61.

Ravi, N., Scaglione, A., and Nedić, A. (2019). A case of distributed optimization in adversarial environment. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5252–5256. doi:10.1109/ICASSP.2019.8683442.

Su, L. and Shahrampour, S. (2018). Finite-time guarantees for byzantine-resilient distributed state estimation with noisy measurements. *CoRR*, abs/1810.10086.

Su, L. and Vaidya, N. (2015a). Byzantine multi-agent optimization: Part I. *CoRR*, abs/1506.04681.

Su, L. and Vaidya, N.H. (2015b). Fault-tolerant distributed optimization (part IV): constrained optimization with arbitrary directed networks. *CoRR*, abs/1511.01821.

Sundaram, S. and Gharesifard, B. (2018). Distributed optimization under adversarial nodes. *IEEE Transactions on Automatic Control*. doi:10.1109/TAC.2018.2836919.

Tsitsiklis, J., Bertsekas, D., and Athans, M. (1986). Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9), 803–812.

Vuković, O. and Dán, G. (2014). Security of fully distributed power system state estimation: Detection and mitigation of data integrity attacks. *IEEE Journal on Selected Areas in Communications*, 32(7), 1500–1508.

Wu, X., Wai, H.T., Scaglione, A., Leshem, A., and Nedić, A. (2018). Data Injection Attack on Decentralized Optimization. In *International Conference on Acoustic Speech and Signal Processing*. IEEE.

Xi, C. and Khan, U.A. (2017). DEXTRA: A fast algorithm for optimization over directed graphs. *IEEE Transactions on Automatic Control*, 62(10), 4980–4993.

Xi, C., Xin, R., and Khan, U.A. (2018). ADD-OPT: Accelerated distributed directed optimization. *IEEE Transactions on Automatic Control*, 63(5), 1329–1339.

Xu, W., Li, Z., and Ling, Q. (2018). Robust decentralized dynamic optimization at presence of malfunctioning agents. *Signal Processing*. doi: 10.1016/j.sigpro.2018.06.024.