INVITED PAPER

Decentralized Zeroth-Order Constrained Stochastic Optimization Algorithms: Frank–Wolfe and Variants With Applications to Black-Box Adversarial Attacks

This article presents an overview of the recent work in the area of distributed zeroth-order optimization, focusing on constrained optimization settings and algorithms built around the Frank–Wolfe framework.

By Anit Kumar Sahu, *Member IEEE*, and Soummya Kar, *Member IEEE*

ABSTRACT | Zeroth-order optimization algorithms are an attractive alternative for stochastic optimization problems, when gradient computations are expensive or when closed-form loss functions are not available. Recently, there has been a surge of activity in utilizing zeroth-order optimization algorithms in myriads of applications including black-box adversarial attacks on machine learning frameworks, reinforcement learning, and simulation-based optimization, to name a few. In addition to utilizing the simplicity of a typical zeroth-order optimization scheme, distributed implementations of zeroth-order schemes so as to exploit data parallelizability are getting significant attention recently. This article presents an overview of recent work in the area of distributed zeroth-order optimization, focusing on constrained optimization settings and algorithms built around the Frank–Wolfe framework. In particular, we review different types of architectures, from master–worker-based decentralized to fully distributed, and describe appropriate zeroth-order projection-free schemes for solving constrained stochastic optimization problems catered to these architectures. We discuss performance issues including convergence rates and dimension dependence. In addition, we also focus on more refined extensions such as by employing variance reduction and describe and quantify convergence rates for a variance-reduced decentralized zeroth-order optimization method inspired by martingale difference sequences. We discuss limitations of zeroth-order optimization frameworks in terms of dimension dependence. Finally, we illustrate the use of distributed zeroth-order algorithms in the context of adversarial attacks on deep learning models.

KEYWORDS | Adversarial learning; distributed optimization; machine learning; stochastic optimization; zeroth-order optimization.

I. INTRODUCTION

The growth of machine learning and data-driven methods over the past few years has largely been fueled by developments in optimization methods which have seen tremendous advances in terms of efficient and flexible access to data and utilization of data, especially in resource-constrained environments and applications. In particular, large-scale training of machine learning models have benefited from the development of scalable effective stochastic optimization methods, where cost gradient information is inexact but unbiased in nature. To address large-scale

Manuscript received December 22, 2019; revised May 17, 2020 and July 24, 2020; accepted July 26, 2020. Date of publication August 18, 2020; date of current version October 27, 2020. The work of Soummya Kar was supported in part by NSF under Grant CCF-1513936. *(Corresponding author: Anit Kumar Sahu.)*

Anit Kumar Sahu is with the Bosch Center for Artificial Intelligence, Pittsburgh, PA 15222 USA (e-mail: anit.sahu@gmail.com).

Soummya Kar is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: soummyak@andrew.cmu.edu).

Digital Object Identifier 10.1109/JPROC.2020.3012609

optimization settings where the data are distributed across different agents, highly scalable stochastic optimization methods using first- [1], [31], [44], [60] and second-order [13], [39], [56], [58] gradient information have been developed and successfully applied to practical settings. While the aforementioned methods provide competitive convergence rates and performance, settings where the explicit form of the loss function is not available or calculating gradients and Hessians is computationally prohibitive, calls for the usage of zeroth-order methods. Also, growing model complexities and the seemingly black-box nature of deployed machine learning models combined with the need to port large-scale models to commodity devices have necessitated to look beyond first- and second-order methods.

Derivative-free optimization or zeroth-order optimization is motivated by settings where the analytical form of the cost function is not directly accessible or when the gradient evaluation is computationally prohibitive. Zeroth-order optimization has found applications in practical scenarios ranging from problems in medical science, material science, and chemistry [10], [18], [36], [37]. More recently, zeroth-order optimization methods have found promising applications in the domain of adversarial machine learning especially in the context of black-box adversarial attacks [3], [5], [6], [23]. While machine learning models have been largely successful in providing state-of-the-art performance in important tasks across the domains of computer vision and natural language processing, they have been found to be fragile in many scenarios. For instance, it has been demonstrated in certain classification problems that associated machine learning models may be readily attacked so as to misclasssify inputs [17]. Designing adversarial attacks, for the purpose of validating and strengthening models, has seen growing interest in the machine learning community [3], [5], [6], [23]. Black-box optimization methods and, in particular, black-box gradient estimation techniques have been particularly effective in black-box attacks [6] on neural networks, that is, in settings where only the model output is known and the architecture and its weights are otherwise unknown. In the context of policy learning and optimization, black-box optimization have also been successfully employed for scalable policy optimization for reinforcement learning [8], for linear quadratic regulators [35], and optimization with bandit feedback [4].

In addition to models getting increasingly complex, the data on which aforementioned models are trained are huge and typically require storage that exceeds the capacity of one machine. Hence, more often than not the data needs to be distributed or is naturally distributed across multiple machines. With the pervasiveness of machine learning models extending to commodity devices such as in the Internet of Things (IoT) context, the data is inherently collected in a distributed manner across devices.[1] The need for stochastic optimization methods that are able to operate without expensive gradient computations and accommodate distributed data across devices calls for the development and understanding of fundamental limits of distributed zeroth-order optimization algorithms which we aim to review in this article.

In this article, we focus on constrained optimization problems where the optimizer has access to (unbiased) loss function evaluations, that is, a zeroth-order oracle [46]. In a constrained optimization setting with access to a zeroth-order oracle, it is natural to develop and employ analogs of projected gradient descent (PGD). However, with biased gradient estimates (the bias resulting from inaccurate gradient estimations from zeroth-order information) and the need for possibly expensive exact projection operations, a naive zeroth-order analog of PGD could suffer from slow convergence and poor dependence on the problem parameters (such as dimension). This article focuses on Frank–Wolfe (FW)-type algorithms [24] and their distributed counterparts. The FW framework provides a unified theme for algorithmic development over different types of optimization architectures. More importantly, zeroth-order analogs of FW discussed in this article provide computational simplicity by avoiding expensive projection operations, instead making use of instances of linear minimizations that can be solved up to a degree of inexactness. Additionally, from an implementation viewpoint, while we seek optimality in general, in applications of interest such as black-box adversarial attacks, a "good enough" feasible point is often adequate (see Section VIII for a relevant illustration), thereby enabling further potential computation savings.

In this article, we discuss both basic and variance reduced zeroth-order stochastic FW algorithms to solve broad classes of smooth constrained optimization problems in both decentralized and distributed setups. By decentralized setups, we mean setups where the devices or workers are connected to a master node to read, write, and exchange information, which is typical in data-center-type environments. By distributed setups, we mean setups where the devices do not have a central coordinator and engage in information exchange in a peer-to-peer manner, by means of local computation and message exchanges with neighboring devices [26], [34], [43], [55], which is typical in IoT-type environments.

We end this section by providing a brief discussion of the main variants of zeroth-order methods to be reviewed in this article, in the decentralized and/or distributed setting. In this article, we first present a setting of the vanilla stochastic FW [2], [52] in which a small batch-size (independent of dimension or the number of iterations) is sampled at each epoch while having access to a zeroth-order oracle at each device. In the vanilla decentralized stochastic zeroth-order FW to circumvent the potential divergence issue due to nondecaying gradient noise and bias, we present a gradient averaging technique used in [38], [49], and [52] to get a surrogate gradient estimate

[1]We use the terms devices, workers, nodes, and agents interchangeably in this article.

which reduces the noise and the associated bias. Intuitively, the gradient averaging technique reduces the linear minimization step in the FW scheme to that of an inexact minimization if the exact gradient was available (more details to follow later). For the variance-reduced decentralized stochastic zeroth-order FW, we demonstrate techniques based on Stochastic Path-Integrated Differential EstimatoR (SPIDER) [14]. We then present a distributed version of the deterministic FW algorithm where a distributed gradient tracking technique is employed. We present rates for both convex and nonconvex loss functions, specifically quantifying the dependence of the primal gap and the FW duality gap on dimension of the optimization problem and the network connectivity. To complement the theoretical results, we also illustrate the efficacy of the presented algorithms through empirical evaluations, by considering the problem of finding universal adversarial perturbations on standard computer vision models.

II. RELATED WORK

For constrained optimization problems, the go to algorithm for most practitioners is the PGD (see [46] for a detailed treatment) and certain variants of it so as to avoid the entire gradient computation at one go. However, more often than not, PGD involves a very expensive projection operation which needs to be solved to a reasonable degree of accuracy or else the algorithm tends to get stuck. Hence, projection-free methods have seen a major surge recently. In the context of projection-free methods, Frank and Wolfe [15] proposed the FW algorithm for smooth convex functions with line search which was then extended in [24] to accommodate inexact linear minimization steps without sacrificing convergence performance. The convergence rates for deterministic FW, that is, with exact gradient information, has been subsequently improved with -additional assumptions in [16] and [29]. Stochastic versions of FW for convex optimization with number of calls to stochastic first-order oracle (SFO) at each iteration dependent on the number of iterations with additional smoothness assumptions have been studied in [20] and [21] so as to obtain faster rates, while Mokhtari *et al.* [38] studied a variant with a mini-batch size of 1. As far as dealing with nonconvex losses is concerned, a deterministic FW algorithm was proposed in [28], while [48], [53], and [62] used variance reduction techniques on a stochastic version of FW and further improved the rates.

Algorithms for zeroth-order convex and nonconvex optimization with access to zeroth-order oracles have been studied in [2], [7], [12], [14], [25], [32], [33], [50], [52], [54], and [59], where the oracles considered are either incremental zeroth-order oracles (IZOs) which returns the exact loss function value at the queried point or stochastic zeroth-order oracles (SZOs) which return an unbiased estimate of the loss function value at the queried point. In particular, Duchi *et al.* [12] established lower bounds for the performance of zeroth-order schemes and the best dimension dependence that can be achieved. In addition to vanilla zeroth-order methods, various forms of variance reduction-based approaches have also been studied in [14], [25], and [33]. However, most of the aforementioned works consider unconstrained optimization problems. With regard to constrained zeroth-order optimization, Balasubramanian and Ghadimi [2], Liu *et al.* [32], and Sahu *et al.* [52] studied the problem for both convex and nonconvex loss functions. In [32], a projection step was considered so as to cater to the constrained problem, while in [52] a gradient smoothing technique was utilized in addition to a stochastic zeroth-order FW framework so as to avoid potentially expensive projection operations. Balasubramanian and Ghadimi [2] used a proximal mapping to further accelerate the convergence of FW algorithm for convex loss functions. Going from centralized to distributed processing, zeroth-order optimization for distributed setups has garnered a lot of interest lately. Unconstrained distributed zeroth-order stochastic optimization with access to an IZO has been studied in [19] and [54], while for a stochastic zeroth oracle was studied in [50]. However, distributed zeroth-order constrained optimization is relatively less explored, which is something we discuss in this article. In this article, we consider the problem of constrained zeroth-order stochastic optimization in both decentralized and distributed scenarios. We constrain the access to information at each node to an SZO and propose a vanilla version and a variance-reduced version so as to improve the iteration dependence even further. For first-order optimization schemes, the primal gap can only be specified in terms of the number of iterations. However, for the zeroth-order schemes, it is crucial to quantify the dependence on the dimension of the problem at hand. In this article, we specifically focus on illustrating the dependence of the primal gap in terms of the number of iterations and the dimension of the optimization problem at hand.

A. Article Organization

Section IV discusses the preliminaries concerning the FW algorithm, black-box gradient estimation, and stochastic FW. Algorithms for decentralized stochastic zeroth-order FW algorithm and its convergence results are presented in Section V. Section VI presents variance-reduction-based algorithms for decentralized stochastic zeroth-order FW algorithm, while in Section VII, algorithms for distributed zeroth-order FW algorithm are discussed and presented. Finally, experimental results are presented in Section VIII, and IX concludes this article. The proofs of technical results skipped in this article can be found in the extended version in [51].

III. STOCHASTIC OPTIMIZATION PRIMER: FIRST AND ZEROTH ORDER

In this section, we briefly review aspects of stochastic and zeroth-order optimization.

A. Optimization Problem

A generic constrained stochastic optimization problem is typically posed as follows:

$$\min_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x}) = \min_{\mathbf{x}\in\mathcal{C}} \mathbb{E}_{\mathbf{y}\sim\mathcal{P}}\left[F(\mathbf{x};\mathbf{y})\right] \tag{1}$$

where $\mathcal{C} \subseteq \mathbb{R}^d$ is the associated constraint. In the machine learning context, the formulation (1) is typically encountered in the context of expected risk minimization where $F(\mathbf{x};\mathbf{y})$ denotes the risk or loss function, depending on the model $\mathbf{x}$ and the data $\mathbf{y}$ obtained from a distribution $\mathcal{P}$ (unknown *a priori*). The stochasticity in the optimization stems from the fact that the functional $F(\cdot\,;\mathbf{y})$ can be queried to obtain gradients or the value of the function itself depending upon the access allowed by the oracle, but the function $f(\mathbf{x})$ may not be queried directly. Thus, be it gradients obtained from an SFO or function values obtained from an SZO, the information obtained is only an unbiased estimate of the gradient $\nabla f(\mathbf{x})$ or the function value $f(\mathbf{x})$. The problem in (1) when applied to a decentralized/distributed setting, that is, a scenario in which the data are distributed across M worker nodes, is often posed as a finite sum problem where $\mathcal{P}$ is taken to be a uniform distribution over $[M] = \{1, 2, \ldots, M\}$ and the goal is to solve a special case of (1)

$$\min_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x}) = \min_{\mathbf{x}\in\mathcal{C}} \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}) \tag{2}$$

where each function f_i is available to the ith node, and each node maintains a local copy of the optimizer $\mathbf{x}$. Furthermore, it can also be assumed that each function $f_i(\cdot)$ is composed of n component functions, that is, the finite sum optimization problem in (25) can be further written as

$$\min_{\mathbf{x}\in\mathcal{C}} \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}) = \min_{\mathbf{x}\in\mathcal{C}} \sum_{i=1}^{M}\sum_{j=1}^{n} f_{i,j}(\mathbf{x}). \tag{3}$$

In such a setting, the data stays at the workers and the task of computing and evaluating the gradients is also relegated to the workers. The master node is tasked with aggregating the stochastic gradients. Throughout this article, we will use all the three versions of the optimization problem at hand and we will make it clear in different settings as to which of these settings we focus on.

B. Zeroth-Order Optimization

In a zeroth-order optimization, information can only be obtained from a zeroth-order oracle which yields evaluations of the loss function at a queried point. Zeroth-order optimization methods are built around gradient estimation schemes from sampled values of the objective function. Different from first-order schemes, the estimated gradient is typically biased.

We briefly describe widely used zeroth-order gradient approximation schemes. The Kiefer–Wolfowitz stochastic approximation (KWSA; see [27]) scheme approximates the gradient by sampling the objective function along the canonical basis vectors. Formally, the gradient estimate can be expressed as

$$\mathbf{g}(\mathbf{x}_t;\mathbf{y}) = \sum_{i=1}^{d} \frac{F\left(\mathbf{x}_t + c_t\mathbf{e}_i;\mathbf{y}\right) - F\left(\mathbf{x}_t;\mathbf{y}\right)}{c_t}\mathbf{e}_i \tag{4}$$

where c_t is a carefully chosen time-decaying sequence and $\{\mathbf{e}_i\}_{i=1}^d$ are the canonical basis vectors and d is the associated dimension of the optimization problem at hand. KWSA requires d samples at each step to evaluate the gradient. However, in order to avoid sampling the objective function d times, random directions-based gradient estimators have been proposed recently (see, e.g. [12] and [47]). The random directions stochastic approximation (RDSA) based gradient estimator involves estimating the directional derivative along a randomly sampled direction from an appropriate probability distribution. Formally, the random directions gradient estimator is given by

$$\mathbf{g}(\mathbf{x}_t;\mathbf{y},\mathbf{z}_t) = \frac{F(\mathbf{x}_t + c_t\mathbf{z}_t;\mathbf{y}) - F(\mathbf{x}_t;\mathbf{y})}{c_t}\mathbf{z}_t \tag{5}$$

where $\mathbf{z}_t \in \mathbb{R}^d$ is a random vector sampled from a probability distribution such that $\mathbb{E}\left[\mathbf{z}_t\mathbf{z}_t^\top\right] = \mathbf{I}_d$ and c_t is a carefully chosen time-decaying sequence. With $c_t \to 0$, both the gradient estimators in (4) and (5) tend to be unbiased estimators of the gradient $\nabla f(\mathbf{x}_t)$. We now turn our attention to zeroth-order constrained stochastic optimization.

C. Zeroth-Order Constrained Stochastic Optimization

In a zeroth-order optimization, with a constraint at hand, a natural way would be to use a stochastic zeroth-order counterpart of PGD. In addition to requiring a possibly expensive projection operation at each step, a zeroth-order PGD takes a step along the estimated zeroth-order gradient aggressively before applying a projection. The aggressive trajectory along the biased zeroth-order gradient might not be a descent direction and the projection operation can also be potentially expensive. The theoretical properties of zeroth-order PGD except for specific constraint sets remain largely unexplored in the literature. In particular, it is not clear as to what convergence rates may be obtained for zeroth-order PGD and their dependence on the number of iterations and the dimension of the problem at hand. Moreover, in applications we consider in this article, that is, black-box adversarial attacks, as it will become clear later in Section VIII, it is often adequate to find a satisfactory feasible point efficiently than obtaining the exact optimal point as long as the feasible point misclassifies most of the examples. It is also

important to understand the dimension dependence of any zeroth-order optimization scheme. Keeping all these factors in mind, we focus on a projection-free, gradient-free stochastic optimization framework and study FW algorithm and its variants in decentralized and distributed settings in this article.

IV. FW: FROM FIRST ORDER TO ZEROTH ORDER

We revisit preliminaries pertaining to the classical FW algorithm next.

A. Background: FW Algorithm

The classical FW algorithm involves approximating the objective by a first-order Taylor approximation. In the case, when exact first-order information is available, that is, one has to access a first-order oracle, a deterministic FW method involves the following steps:

$$\begin{aligned}\mathbf{v}_t &= \operatorname*{argmin}_{\mathbf{v}\in\mathcal{C}}\langle\nabla f(\mathbf{x}_t),\mathbf{v}\rangle\\ \mathbf{x}_{t+1} &= (1-\gamma_{t+1})\,\mathbf{x}_t+\gamma_{t+1}\mathbf{v}_t\end{aligned}\tag{6}$$

where $\gamma_t = 2/(t+2)$. At every epoch, a linear minimization oracle (LMO) is queried. The minimization in (6) is a linear program when $\mathcal{C}$ is given by linear constraints and can be performed efficiently without much computational overload. More generally, a wide class of structured constraint sets (see [24, Table 1]) enable low computational complexity minimizations for (6). Equivalently, primal–dual methods can be used in high-dimensional settings, which still require an exact projection which can be computationally taxing. It is worth noting that the exact minimization in (6) can be replaced by an inexact minimization of the following form, where a $\mathbf{v}\in\mathcal{C}$ is chosen to satisfy

$$\langle\nabla f(\mathbf{x}_t),\mathbf{v}\rangle \le \operatorname*{argmin}_{\mathbf{v}\in\mathcal{C}}\langle\nabla f(\mathbf{x}_t),\mathbf{v}\rangle+\gamma_t C_1$$

where C_1 is a positive constant and the algorithm can be shown to retain the same convergence rate (see, e.g., [24]). The room for inexactness for the minimization in (6) and it admitting to computationally efficient solutions for widely used structured constraint sets makes FW an attractive avenue for the class of optimization problems considered in this article.

Before getting into the stochastic case, we demonstrate how a typical zeroth-order FW framework corresponds to an inexact classical FW optimization in the deterministic setting.

B. Deterministic Zeroth-Order FW

The deterministic version of the optimization in (1) can be restated as follows:

$$\min_{\mathbf{x}\in\mathcal{C}} F(\mathbf{x}).\tag{7}$$

We state the main assumptions that we will be making use of throughout this article.

Assumption A1: In problem (1), the set $\mathcal{C}$ is bounded with finite diameter R.

Assumption A2: f_i's are Lipschitz continuous with $\left(\mathbb{E}\left[\|\nabla_x f_i(\mathbf{x};\cdot)\|^2\right]\right)^{1/2}\le L_1$ for all $\mathbf{x}\in\mathcal{C}$.

Assumption A3: The function f is continuously differentiable and is L-smooth, that is, its gradient ∇f is L-Lipschitz continuous over the set $\mathcal{C}$, that is, for all $x,y\in\mathcal{C}$,

$$\|\nabla f(\mathbf{x})-\nabla f(\mathbf{y})\|\le L\|\mathbf{x}-\mathbf{y}\|.\tag{8}$$

Assumption A4: $\mathbf{z}_t$'s are drawn from a distribution μ such that $M(\mu)=\mathbb{E}\left[\|\mathbf{z}_t\|^6\right]$ is finite, and for any vector $\mathbf{g}\in\mathbb{R}^d$, there exists a function $s(d):\mathbb{N}\mapsto\mathbb{R}_+$ such that

$$\mathbb{E}\left[\|\langle\mathbf{g},\mathbf{z}_t\rangle\mathbf{z}_t\|^2\right]\le s(d)\|\mathbf{g}\|^2.$$

Assumption A5: The gradient estimates $\nabla F_i(\mathbf{x};\mathbf{y})$ of $\nabla f_i(\mathbf{x})$ are unbiased, that is, $\mathbb{E}_{\mathbf{y}\sim\mathcal{P}_i}[\nabla F_i(\mathbf{x};\mathbf{y})]=\nabla f_i(\mathbf{x}),\ \forall i=1,\dots,M$ and satisfy

$$\mathbb{E}\left[\|\nabla F_i(\mathbf{x};\mathbf{y})-\nabla f_i(\mathbf{x})\|^2\right]\le\sigma^2\quad\forall i.\tag{9}$$

In what follows, we demonstrate the equivalence of a typical zeroth-order FW framework and that of an inexact classical FW optimization. We consider KWSA for gradient estimation for this purpose. In particular, the KWSA gradient estimator in (4) can be expressed as follows:

$$\begin{aligned}\mathbf{g}(\mathbf{x}_t) &= \sum_{i=1}^{d}\frac{F(\mathbf{x}_t+c_t\mathbf{e}_i)-F(\mathbf{x}_t)}{c_t}\mathbf{e}_i\\ &= \nabla F(\mathbf{x}_t)+\sum_{i=1}^{d}\frac{c_t}{2}\langle\mathbf{e}_i,\nabla^2F(\mathbf{x}_t+\lambda c_t\mathbf{e}_i)\mathbf{e}_i\rangle\mathbf{e}_i\end{aligned}\tag{10}$$

where we use Taylor's theorem to obtain a second-order representation of the gradient and $\lambda\in[0,1]$. The linear optimization step then simplifies to

$$\begin{aligned}\langle\mathbf{v},\mathbf{g}(\mathbf{x}_t)\rangle &= \langle\mathbf{v},\nabla F(\mathbf{x}_t)\rangle\\ &\quad+\frac{c_t}{2}\sum_{i=1}^{d}\langle\mathbf{e}_i,\nabla^2F(\mathbf{x}_t+\lambda c_t\mathbf{e}_i)\mathbf{e}_i\rangle\langle\mathbf{v},\mathbf{e}_i\rangle\\ &\Rightarrow \min_{\mathbf{v}\in\mathcal{C}}\langle\mathbf{v},\mathbf{g}(\mathbf{x}_t)\rangle\le\min_{\mathbf{s}\in\mathcal{C}}\langle\mathbf{s},\nabla F(\mathbf{x}_t)\rangle+\frac{c_tLRd}{2}\end{aligned}\tag{11}$$

where R is the diameter of the constraint set $\mathcal{C}$. In particular, on choosing c_t and γ_t to be (γ_t/d) and $2/(t+1)$, respectively, we obtain the following bound characterizing the primal gap.

Theorem 1 [52]: Assume the function $F(\cdot)$ is convex. Let Assumptions A1–A3 hold. Given the zeroth-order FW

Algorithm 1 Deterministic Zeroth-Order FW

Require: Input, Loss Function $F(x)$, L (Lipschitz constant for the gradients), Convex Set $\mathcal{C}$, Sequences $\gamma_t = \frac{2}{t+1}, c_t = \frac{L\gamma_t}{d}$.

Output: $\mathbf{x}_T$ or $\frac{1}{T}\sum_{t=1}^{T}\mathbf{x}_t$.

1: Initialize $\mathbf{x}_0 \in \mathcal{C}$
2: **for** $t = 0, 1, \ldots, T-1$ **do**
3: Compute $\mathbf{g}(\mathbf{x}_t) = \sum_{i=1}^{d} \frac{F(\mathbf{x}_t + c_t\mathbf{e}_i) - F(\mathbf{x}_t)}{c_t}\mathbf{e}_i$,
4: Compute $\mathbf{v}_t = \operatorname{argmin}_{\mathbf{s}\in\mathcal{C}}\langle \mathbf{s}, \mathbf{g}(\mathbf{x}_t)\rangle$,
5: Compute $\mathbf{x}_{t+1} = (1-\gamma_t)\,\mathbf{x}_t + \gamma_t\mathbf{v}_t$.
6: **end for**

algorithm in Algorithm 1, we obtain the following bound:

$$F(\mathbf{x}_t) - F(\mathbf{x}^*) \leq \frac{Q_{ns}}{t+2} \tag{12}$$

where $Q_{ns} = \max\{2(F(\mathbf{x}_0) - F(\mathbf{x}^*)), 4LR^2\}$.

In summary, Theorem 1 shows the equivalence of the deterministic zeroth-order FW algorithm and that of the inexact classical FW algorithm with the primal gap being dimension-independent. The dimension independence comes at the cost of queries to the zeroth-order oracle which scales linearly with dimension. In the sequel, we will focus on the random directions gradient estimator in (5) for the stochastic zeroth-order FW algorithm.

C. Stochastic FW

In the classical FW algorithm, the replacement of the true gradient $\nabla f(\mathbf{x}_k)$ by its stochastic counterpart, that is, $\nabla F(\mathbf{x}_k; \mathbf{y}_k)$ could make the algorithm divergent due to an additional variance term due to the gradient approximations. In addition to the potential divergence, the usage of an unbiased estimate of the gradient makes the LMO constraint to hold only in expectation. This particular issue is further exacerbated due to biased gradient estimates. We use a well-known averaging trick (see, e.g., [52]) to counter this problem which is as follows:

$$\mathbf{d}_t = (1-\rho_t)\,\mathbf{d}_{t-1} + \rho_t\mathbf{g}\left(\mathbf{x}_t, \mathbf{y}_t\right) \tag{13}$$

where $\mathbf{g}\left(\mathbf{x}_t, \mathbf{y}_t\right)$ is a gradient approximation, $\mathbf{d}_0 = \mathbf{0}$ and ρ_t is a time-decaying sequence. The gradient smoothing scheme allows for $\mathbb{E}\left[\|\mathbf{d}_t - \nabla f\left(\mathbf{x}_t\right)\|^2\right]$ to go to zero asymptotically. With the above averaging scheme, we replace the linear minimization and the subsequent steps as follows:

$$\begin{aligned}
\mathbf{d}_t &= (1-\rho_t)\,\mathbf{d}_{t-1} + \rho_t\mathbf{g}\left(\mathbf{x}_t, \mathbf{y}_t\right) \\
\mathbf{v}_t &= \operatorname*{argmin}_{\mathbf{v}\in\mathcal{C}}\langle \mathbf{d}_t, \mathbf{v}\rangle \\
\mathbf{x}_{t+1} &= (1-\gamma_{t+1})\,\mathbf{x}_t + \gamma_{t+1}\mathbf{v}_t.
\end{aligned} \tag{14}$$

Before proceeding to the algorithms, we briefly describe an improvised gradient estimation technique to be used in the context of zeroth-order optimization schemes. In addition to the schemes, as outlined in (4) and (5), we employ a gradient estimator (I-RDSA) by sampling m-directions $\{\mathbf{z}_{i,t}\}_{i=1}^{m}$ at each time followed by averaging, that is,

$$\begin{aligned}
&\mathbf{g}_m(\mathbf{x}_t; \mathbf{y}_t, \mathbf{z}_{i,t}) \\
&\quad = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{F\left(\mathbf{x}_t + c_t\mathbf{z}_{i,t}; \mathbf{y}\right) - F\left(\mathbf{x}_t; \mathbf{y}\right)}{c_t}\mathbf{z}_{i,t}\right).
\end{aligned} \tag{15}$$

The above scheme computes gradient estimates in m-directions before averaging them to somewhat counter the variance. However, m is chosen to be independent of d so that the query complexity does not scale with dimension. In order to quantify the benefits of using such a scheme, we present the statistics concerning the gradient approximation of RDSA and I-RDSA. We have from [12] for RDSA

$$\begin{aligned}
&\mathbb{E}_{\mathbf{z}_t\sim\mu, \mathbf{y}_t\sim\mathcal{P}}\left[\mathbf{g}(\mathbf{x}; \mathbf{y}_t, \mathbf{z}_t)\right] \\
&\quad = \nabla f\left(\mathbf{x}\right) + c_t L\mathbf{v}\left(\mathbf{x}, c_t\right) \\
&\mathbb{E}_{\mathbf{z}_t\sim\mu, \mathbf{y}_t\sim\mathcal{P}}\left[\|\mathbf{g}(\mathbf{x}; \mathbf{y}_t, \mathbf{z}_t)\|^2\right] \\
&\quad \leq 2s(d)\mathbb{E}\left[\|\nabla F(\mathbf{x}; \mathbf{y}_t)\|^2\right] + \frac{c_t^2}{2}L^2M(\mu).
\end{aligned} \tag{16}$$

Using (16), similar statistics for the improvised RDSA gradient estimator can be evaluated as follows:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{z}_t\sim\mu, \mathbf{y}_t\sim\mathcal{P}}\left[\mathbf{g}_m(\mathbf{x}; \mathbf{y}_t, \mathbf{z}_t)\right] \\
&= \nabla f\left(\mathbf{x}\right) + \frac{c_t}{m}L\mathbf{v}\left(\mathbf{x}, c_t\right) \\
&\mathbb{E}_{\mathbf{z}_t\sim\mu, \mathbf{y}_t\sim\mathcal{P}}\left[\|\mathbf{g}_m(\mathbf{x}; \mathbf{y}_t, \mathbf{z}_t)\|^2\right] \\
&\leq \left(\frac{1+m}{2m}\right)c_t^2L^2M(\mu) + 2\left(1 + \frac{s(d)}{m}\right)\mathbb{E}\left[\|\nabla F(\mathbf{x}; \mathbf{y}_t)\|^2\right]
\end{aligned} \tag{17}$$

where $\|\mathbf{v}\left(\mathbf{x}, c_t\right)\| \leq 1/2\mathbb{E}\left[\|\mathbf{z}\|^3\right]$. A proof for (17) can be found in [32]. As we will see later, the I-RDSA scheme improves the dimension dependence of the primal gap, but it comes at the cost of m calls to the SZO. We are now ready to discuss zeroth-order stochastic optimization in decentralized settings.

V. DECENTRALIZED STOCHASTIC CONSTRAINED ZEROTH-ORDER OPTIMIZATION

For a decentralized setting, the network architecture typically consists of workers and a master node. We specifically assume that there are M workers and one master node. In this scenario, we assume that the data are distributed across all the workers which in turn are all drawn from possibly different data distributions and the workers try to solve a decentralized counterpart of the optimization problem in (1). In essence, we address the following

Algorithm 2 Decentralized Stochastic Gradient Free FW

Require: Input, Loss Function $F(x;y)$, Convex Set $\mathcal{C}$, number of directions m, sequences $\gamma_t = \frac{2}{t+8}$,

$$(\rho_t, c_t)_{I-RDSA} = \left(\frac{4}{\left(1+\frac{d}{m}\right)^{1/3}(t+8)^{2/3}}, \frac{2\sqrt{m}}{d^{3/2}(t+8)^{1/3}} \right)$$

Output: $\mathbf{x}_T$ or $\frac{1}{T}\sum_{t=0}^{T-1}\mathbf{x}_t$.

1: Initialize $\mathbf{x}_0 \in \mathcal{C}$
2: **for** $t = 0, 2, \ldots, T-1$ **do**
3: At each worker i compute
I-RDSA: Sample $\{\mathbf{z}_{n,t}\}_{n=1}^m \sim \mathcal{N}(0, \mathbf{I}_d)$
$\mathbf{g}_i(\mathbf{x}_t;\mathbf{y}) = \frac{1}{m}\sum_{n=1}^m \frac{F(\mathbf{x}_t + c_t\mathbf{z}_{n,t};\mathbf{y}) - F(\mathbf{x}_t;\mathbf{y})}{c_t}\mathbf{z}_{n,t}$
4: Workers compute $\mathbf{g}_{i,t} = (1-\rho_t)\mathbf{g}_{i,t-1} + \rho_t \mathbf{g}_i(\mathbf{x}_t, \mathbf{y})$
5: Push $\mathbf{g}_{i,t}$ to the master node.
6: Master node computes $\mathbf{g}_t = \frac{1}{M}\sum_{i=1}^M \mathbf{g}_{i,t}$.
7: Master node computes $\mathbf{v}_t = \operatorname{argmin}_{\mathbf{s}\in\mathcal{C}}\langle \mathbf{s}, \mathbf{g}_t\rangle$,
8: Master node computes $\mathbf{x}_{t+1} = (1-\gamma_t)\mathbf{x}_t + \gamma_t \mathbf{v}_t$ and sends it to all nodes.
9: **end for**

problem:

$$\min_{\mathbf{x}\in\mathcal{C}} \left\{ f(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^M f_i(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^M \mathbb{E}_{\mathbf{y}\sim\mathcal{P}_i}\left[F(\mathbf{x};\mathbf{y})\right] \right\} \tag{18}$$

where $\mathcal{P}_i$ is the local data distribution at node i.

We specifically employ Algorithm 2 to solve the optimization problem in (18) which we briefly describe below. At each time, each worker samples a data point and corresponding Gaussian random vectors to estimate the gradient using I-RDSA. Upon evaluation of the gradient locally, each worker uses gradient smoothing before sending it to the master node. Upon receiving the smoothed gradient estimates from the workers, the master node averages them and computes the next iterate and sends the new iterate to the worker nodes.

The study of the primal gap for convex and nonconvex loss functions is based on the evolution of the mean square error (mse) of the surrogate gradient term that we track through the gradient averaging technique. We quantify the primal gap for convex loss functions next.

Theorem 2 (Convex Losses [52]): Let Assumptions A1–A4 hold and let each f_i be L-smooth. Let the sequence γ_t be given by $\gamma_t = 2/(t+8)$. In order to achieve a primal suboptimality gap of ϵ for convex loss functions in Algorithm 2, that is,

$$\mathbb{E}\left[f(\mathbf{x}_t) - f(\mathbf{x}^*)\right] \leq \epsilon, \tag{19}$$

the number of queries to the SZO is given by $O\left(d/\epsilon^3\right)$.
We provide a brief proof sketch of Theorem 2.

Theorem 2 establishes the dimension dependence of the primal gap to be $d^{1/3}$. The iteration dependence, that is, $O(T^{-1/3})$ matches that of the stochastic FW with first-order information as in [38]. The dimension dependence for the number of queries cannot be improved as it matches the minimax lower bound in [12]. Denote the FW duality gap by

$$\mathcal{G}(\mathbf{x}) = \max_{\mathbf{v}\in\mathcal{C}}\langle \nabla F(\mathbf{x}), \mathbf{x} - \mathbf{v}\rangle.$$

The proof is built around [52, Lemma 3.3], which characterizes the evolution of the primal gap for the function and the main result of the Lemma can be stated as follows:

$$\begin{aligned} f(\mathbf{x}_{t+1}) - f(\mathbf{x}^*) \leq (1-\gamma_{t+1})(f(\mathbf{x}_t) - f(\mathbf{x}^*)) \\ + \gamma_{t+1} R\|\nabla f(\mathbf{x}_t) - \mathbf{g}_t\| + \frac{LR^2\gamma_{t+1}^2}{2}. \end{aligned} \tag{20}$$

The performance of the pseudo-gradient estimate generated at each worker node i can be characterized by

$$\begin{aligned} &\mathbb{E}\left[\|\nabla f_i(\mathbf{x}_t) - \mathbf{g}_{i,t}\|^2\right] \\ &\quad \leq 2\rho_t^2\left(\sigma^2 + 2L_1^2\right) \\ &\qquad + \frac{\rho_t}{2m^2}c_t^2 L^2 M(\mu) + 8\rho_t^2\left(1 + \frac{s(d)}{m}\right)L_1^2 \\ &\qquad + \left(\frac{1+m}{2m}\right)\rho_t^2 c_t^2 L^2 M(\mu) + \frac{2L^2R^2\gamma_t^2}{\rho_t} \\ &\qquad + \left(1 - \frac{\rho_t}{2}\right)\mathbb{E}\left[\|\nabla f_i(\mathbf{x}_{t-1}) - \mathbf{g}_{i,t-1}\|^2\right] \end{aligned} \tag{21}$$

and thus a bound for $\mathbb{E}\left[\|\nabla f(\mathbf{x}_t) - \mathbf{g}_t\|^2\right]$ is obtained by noting that

$$\mathbb{E}\left[\|\nabla f(\mathbf{x}_t) - \mathbf{g}_t\|^2\right] \leq \frac{1}{M}\sum_{i=1}^M \mathbb{E}\left[\|\nabla f_i(\mathbf{x}_t) - \mathbf{g}_{i,t}\|^2\right].$$

Finally, to obtain bounds and characterize the recursions above, we use [52, Lemma B.1] to obtain

$$\mathbb{E}\left[\|\nabla f(\mathbf{x}_t) - \mathbf{g}_t\|^2\right] \leq \frac{2Q}{(t+8)^{2/3}} \tag{22}$$

where Q is a constant characterized by the different algorithm parameters L, R, σ^2, d, and m. Plugging (22) to (20) provides the result needed.

For convex losses, the duality gap is given by

$$\mathbb{E}\left[\min_{t=0,\ldots,T-1}\mathcal{G}(\mathbf{x}_t)\right] = O\left(T^{-1/3}\right). \tag{23}$$

The duality gap characterization is obtained by using the following inequality for the duality gap from [52]:

$$\begin{aligned} \gamma\mathbb{E}\left[\mathcal{G}(\mathbf{x}_t)\right] \leq \mathbb{E}\left[f(\mathbf{x}_t)\right] - \mathbb{E}\left[f(\mathbf{x}_{t+1})\right] \\ + \gamma R\mathbb{E}\left[\|\nabla f(\mathbf{x}_t) - \mathbf{g}_t\|\right] + \frac{LR^2\gamma^2}{2} \end{aligned}$$

where for $\mathbb{E}\left[\|\nabla f(\mathbf{x}_t) - \mathbf{g}_t\|\right]$, we use the relation in (22).

Technically speaking, while it is important to characterize the primal gap to benchmark the performance, it is important to characterize the duality gap as it is used as a stopping criterion for FW algorithms. The FW duality gap can be difficult to compute in certain scenarios. However, for a large class of constraint sets arising in practice in which the linear minimization can be reduced to a linear program (that is amenable to efficient solutions), the FW duality gap can be effectively computed and used as a stopping criterion. For other scenarios, an approximate evaluation of the duality gap would also suffice, that is, solving the minimization approximately.

We quantify the performance of the algorithm for nonconvex loss functions using FW duality gap. The usage of FW duality gap to characterize the convergence for nonconvex loss functions is standard in the literature (see, e.g., [48]). Also note that when $\mathcal{G}(\mathbf{x}) = 0$, it also ensures that $\mathbf{x}$ is a stationary point to the loss function. Thus, $\mathcal{G}(\mathbf{x})$ may be regarded as a measure of stationarity for the iterate $\mathbf{x}$.

Theorem 3 (Nonconvex Losses [52]): Let Assumptions A1–A4 hold and let each f_i be L-smooth. Let $\gamma_t = T^{-3/4}\ \forall t$. Then, in order to obtain a duality gap of ϵ for nonconvex loss functions in Algorithm 2, that is,

$$\mathbb{E}\left[\min_{t=0,\ldots,T-1} \mathcal{G}(\mathbf{x}_t)\right] \leq \epsilon \tag{24}$$

the number of queries to the SZO is given by $O(d^{4/3}/\epsilon^4)$.

Theorem 3 establishes the dimension dependence of the FW duality gap for nonconvex losses to be $d^{4/3}$ and has the same iteration dependence, that is, $O(T^{-1/4})$ as that of the algorithm in [48].

While the iteration dependence matches that of standard first-order methods, the biggest bottleneck for the above approach is the gradient averaging technique and the mse of the gradient estimate. At this point, we ask a pertinent question whether the mse performance of the gradient estimation can be improved, which leads us to Section VI, where we use SPIDER [14], a variance reduction technique to further improve the performance.

VI. DECENTRALIZED VARIANCE-REDUCED STOCHASTIC FW

In this section, for the optimization problem in (1), we further enforce that $\mathcal{P}$ is a uniform distribution over $[M] = \{1, 2, \ldots, M\}$ and the goal is to solve a special case of (1):

$$\min_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x}) = \min_{\mathbf{x}\in\mathcal{C}} \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}) \tag{25}$$

where each function f_i is available to the ith node and each node maintains a copy of the optimizer $\mathbf{x}$. To further illustrate stochasticity in the objective function, we assume that each function $f_i(\cdot)$ is composed of n component functions, that is, the finite sum optimization problem in (25) can be further written as

$$\min_{\mathbf{x}\in\mathcal{C}} \left\{ f(x) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}) = \sum_{i=1}^{M}\sum_{j=1}^{n} f_{i,j}(\mathbf{x}) \right\}. \tag{26}$$

We employ the SPIDER variance reduction technique, in particular, to address this problem.

The major difference with the stochastic setting is that a suitably and carefully crafted variance reduction technique can be used so as to reduce query complexity. Recently, Fang *et al.* [14] proposed the SPIDER technique, for unconstrained optimization in centralized settings for both first-order and zeroth-order oracles. In this article, we generalize SPIDER to the constrained decentralized zeroth-order setting.

A. Stochastic Path-Integrated Differential Estimator

SPIDER is built for dynamic tracking, while avoiding excessive querying to oracles and ultimately reducing query complexity. We provide a brief overview of SPIDER before proceeding further and refer the reader to [14] for a detailed treatment of SPIDER.

Given an observed sequence $x_{0:K}$, we want to track a sequence $Q(x_k)$ for $k = 0, \ldots, K$ with the assumption that an initial estimate of $Q(x_0)$, $\widehat{Q}(x_0)$ is available. Furthermore, an unbiased estimator $\zeta_k(x_{0:k})$ of $Q(x_k) - Q(x_{k-1})$ is also available such that

$$\mathbb{E}\left[\zeta_k(x_{0:k})\,|x_{0:k}\right] = Q(x_k) - Q(x_{k-1}).$$

Then, the estimator $\widehat{Q}(x_{0:K}) = \widehat{Q}(x_0) + \sum_{k=1}^{K} \zeta_k x_{0:k}$ is the integrated (in the discrete sense) stochastic differential estimate. This estimator, when written in a recursive form, can be shown to reduce variance. It turns out that SPIDER can be used to track many quantities of interest, such as stochastic gradient, loss function values, and zeroth-order gradient estimates.

Let $q \in \mathbb{N}_+$ denote a period parameter. At the beginning of each period, that is, $\text{mod}(t, q) = 0$, each worker computes gradient estimates of all of its local component functions using KWSA. It is noteworthy that KWSA is the most query hungry scheme, but at the same time it is very accurate. The master node takes the average of all such averaged gradient received from all workers and computes the next iterate and broadcasts it to all workers. At other times, the workers select a mini-batch of local component functions and use RDSA to estimate and update the gradients which is then sent to the master node. Then, the master calculates the average of the M signals and computes the next iterate and broadcasts it to all workers. The full description of our proposed decentralized variance-reduced zeroth-order FW is outlined in Algorithm 3.

Algorithm 3 Decentralized Variance-Reduced Zeroth-Order FW

Require: Input, Loss Function $f(x)$, Convex Set $\mathcal{C}$, period q, Sample Sizes S_1, S_2

Output: $\mathbf{x}_T$.

1: Initialize $\mathbf{x}_0 \in \mathcal{C}$
2: **for** $t = 0, 2, \ldots, T-1$ **do**
3: At each worker i compute
4: **if** mod(t,q)=0
5: Draw $S_1' = S_1/Md$ samples for each dimension at each worker i and compute its local gradient $\mathbf{e}_j^\top \mathbf{g}_i(\mathbf{x}_t) = \frac{1}{n}\sum_{j=1}^{n} \frac{f_{i,j}(\mathbf{x}_t+\eta\mathbf{e}_j)-f_{i,j}(\mathbf{x}_t)}{\eta}$ along each canonical basis vector $\mathbf{e}_j$.
6: Each worker updates $\mathbf{g}_{i,t} = \mathbf{g}_i(\mathbf{x}_t)$
7: **else**
8: Draw S_2 pairs of component functions and Gaussian random vectors $\{\mathbf{z}\}$ at each worker i and update

$$\mathbf{g}_i(\mathbf{x}_t) = \frac{1}{|S_2|}\sum_{j\in S_2} \frac{f_{i,j}(\mathbf{x}_t+\eta\mathbf{z}) - f_{i,j}(\mathbf{x}_t)}{\eta}\mathbf{z} - \frac{f_{i,j}(\mathbf{x}_{t-1}+\eta\mathbf{z}) - f_{i,j}(\mathbf{x}_{t-1})}{\eta}\mathbf{z}$$

9: Each worker updates $\mathbf{g}_{i,t} = \mathbf{g}_i(\mathbf{x}_t) + \mathbf{g}_{i,t-1}$
10: **end if**
11: Each worker pushes $\mathbf{g}_{i,t}$ to the master node.
12: Master node computes $\mathbf{g}_t = \frac{1}{M}\sum_{i=1}^{M} \mathbf{g}_{i,t}$ and sends it to all workers.
13: Master node computes $\mathbf{v}_t = \operatorname{argmin}_{\mathbf{s}\in\mathcal{C}}\langle \mathbf{s}, \mathbf{g}_t\rangle$,
14: Master node computes $\mathbf{x}_{t+1} = (1-\gamma_t)\,\mathbf{x}_t + \gamma_t \mathbf{v}_t$ and sends it to all workers.
15: **end for**

Intuitively speaking, the algorithm combines a query hungry yet accurate gradient estimation with a query efficient potentially inaccurate gradient estimation and tracking so as to reduce query complexity. In the sequel, we characterize the query complexity, the gradient tracking mse in terms of ϵ primal gap for convex losses and ϵ FW duality gap for nonconvex losses. The major improvement of the variance reduction scheme is in terms of the gradient tracking performance, that is, with $S_2 = ((2d+9)\sqrt{n}/n_0)$, $q = (n_0\sqrt{n}/6)$, where $n_0 \in [1, (\sqrt{n}/6)]$, the mse of the tracked gradient error is given by

$$\mathbb{E}\left[\|\nabla f(\mathbf{x}_t) - \mathbf{g}_t\|^2\right] \leq \frac{\epsilon^2}{3}. \tag{27}$$

With the gradient estimation performance in place, we now characterize the performance of Algorithm 3.

Theorem 4: Let Assumptions A1–A4 hold and let us consider Algorithm 3 with $S_2 = ((2d+9)\sqrt{Mn})/n_0$, $q = (n_0\sqrt{Mn})/6$, where $n_0 \in [1, \sqrt{Mn}/6]$, $\gamma_t = O(\epsilon)$, and $S_1 = Mnd$, we have that the number of queries required to obtain an FW duality gap of ϵ, that is,

$$\mathbb{E}\left[\min_{t=0,\ldots,T-1} \mathcal{G}(\mathbf{x}_t)\right] \leq \epsilon$$

is given by $O\left(\min\{d\sqrt{n}/\epsilon^2, d/\epsilon^3\}\right)$.

The proof of Theorem 4 follows similarly as that of the proof of Theorem 3 and the gradient approximation bound in (27) is as derived in [15, Lemma 11].

For a fair comparison with the stochastic case, the query complexity obtained for the finite sum case can be reproduced by $S_2 = (30(2d+9)\sigma)/(n_0\epsilon)$, $q = (5n_0\sigma)/\epsilon$, where $n_0 \in [1, \sqrt{Mn}/6]$, $\gamma_t = O(\epsilon)$ and $S_1 = (96d\sigma^2)/\epsilon^2$. Theorem 4 improves the iteration dependence even further from that of the vanilla decentralized stochastic FW in the regime where $Mn \geq \epsilon^{-2}$. However, the dimension dependence may not be improved any further.

In the sequel, we will focus on fully distributed settings where the network architecture is devoid of a master node or a fusion center.

VII. DISTRIBUTED ZEROTH-ORDER FW

In this section, we study the finite sum version of the optimization problem at hand, that is,

$$\min_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x}) = \min_{\mathbf{x}\in\mathcal{C}} \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}) \tag{28}$$

in a network of M agents, where the ith agent has access to the ith function. In this section, we consider a fully distributed setup which is devoid of a central coordinator. Also, different from the rest of this article, we consider a deterministic zeroth-order optimization problem, as depicted in (28). In order to exchange information, the nodes use peer-to-peer message passing while conforming to a prespecified possibly sparse communication graph. Unlike the decentralized setting, the individual nodes are not synchronized in terms of the gradient and the secant direction obtained from the LMO. The internode communication network to which the information exchange between nodes conforms to is modeled as an undirected simple connected graph $G = (V, E)$, with $V = [1, \ldots, M]$ and E denoting the set of nodes and communication links. The neighborhood of node n is given by $\Omega_n = \{l \in V \,|\, (n,l) \in E\}$. The node n has degree $d_n = |\Omega_n|$. The structure of the graph is described by the $M \times M$ adjacency matrix, $\mathbf{A} = \mathbf{A}^\top = [\mathbf{A}_{ij}]$, $\mathbf{A}_{ij} = 1$, if $(i,j) \in E$, $\mathbf{A}_{ij} = 0$, otherwise. The graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is positive semidefinite, with eigenvalues ordered as $0 = \lambda_1(\mathbf{L}) \leq \lambda_2(\mathbf{L}) \leq \cdots \leq \lambda_M(\mathbf{L})$, where $\mathbf{D}$ is given by $\mathbf{D} = \operatorname{diag}(d_1 \cdots d_M)$. Moreover, for a connected graph, $\lambda_2(\mathbf{L}) > 0$ (see [9]).

At every time instant t, an agent i exchanges its current iterate with its neighbors and averages the iterates as

follows:

$$\bar{\mathbf{x}}_t^i \leftarrow \sum_{j=1}^{M} W_{ij}\mathbf{x}_t^j.$$

The weights W_{ij} are collected in a matrix $\mathbf{W}$. One of the ways in which the weight matrix $\mathbf{W}$ can be designed is as follows: $\mathbf{W} = \mathbf{I} - \delta\mathbf{L}$, where $\mathbf{L}$ is the graph Laplacian. At the new averaged iterate, that is, $\bar{\mathbf{x}}_t^i$, each agent i computes its local gradient estimate $\mathbf{g}_t^i$ by using KWSA. As the first alternative, the agents could just exchange the gradient estimates among themselves. However, in such a case to ensure that the difference of the gradient estimate at each agent is $O(\gamma_t)$ close in ℓ_2 norm to the network-averaged gradient would take $O(\log t)$ rounds of communication. This observation follows from the fact that

$$\sqrt{\sum_{i=1}^{M}\left\|\sum_{j=1}^{M} W_{ij}\mathbf{g}_t^j - \bar{\mathbf{g}}_t\right\|^2} \leq |\lambda_2(\mathbf{W})|\sqrt{\sum_{i=1}^{M}\|\mathbf{g}_t^i - \bar{\mathbf{g}}_t\|^2}.$$

Hence, we resort to a gradient tracking approach so as to ensure that only one round of communication is required. Thus, to closely replicate the centralized gradient, the agents then average the gradient estimates locally in their neighborhood while using gradient tracking as follows:

$$\mathbf{G}_t^j = \bar{\mathbf{g}}_{t-1}^i + \mathbf{g}_t^i - \mathbf{g}_{t-1}^i$$

where $\mathbf{g}_t^i$ is computed using KWSA, that is,

$$\mathbf{g}_t^i = \sum_{j=1}^{d} \frac{f_i\left(\bar{\mathbf{x}}_t^i + c_t\mathbf{e}_j\right) - f_i(\bar{\mathbf{x}}_t^i)}{c_t}\mathbf{e}_j$$

where $c_t = \gamma_t/d$ and $\mathbf{e}_j$ denotes the jth canonical basis vector. Note that the agents query a zeroth-order oracle here to obtain the gradient estimate. Subsequently, the agents exchange the gradient estimates as follows:

$$\bar{\mathbf{g}}_t^i \leftarrow \sum_{j=1}^{M} W_{ij}\mathbf{G}_t^j. \tag{29}$$

Finally, each node i implements the FW step

$$\mathbf{x}_{t+1}^i \leftarrow (1-\gamma_t)\bar{\mathbf{x}}_t^i + \gamma_t\mathbf{v}_t^i, \quad \text{where } \mathbf{v}_t^i \in \arg\min_{\mathbf{v}\in\mathcal{C}} \left\langle \bar{\mathbf{g}}_t^i, \mathbf{v}\right\rangle.$$

It is worth noting that unlike other distributed optimization protocols such as distributed SGD which involve just one round of communication, a distributed FW algorithm requires two rounds of communication: one for the iterate and one for the gradient estimate. Without exchanging iterates, it would take more rounds of communications at each time step to ensure that the pseudo-gradient estimates are close to each other. Similarly, without exchanging the pseudo-gradient estimates $\mathbf{G}_t^j$, the secant direction $\mathbf{v}_t^i$ could be possibly very different at different nodes, thereby leading to slow convergence. The algorithm is summarized in Algorithm 4.

Algorithm 4 Distributed Zeroth-Order FW

1: **Input**: Initial point $\mathbf{x}_1^i$ for $i = 1, \ldots, M$.
2: **for** $t = 1, 2, \ldots$ **do**
3: *Consensus*: approximate the average iterate:

$$\bar{\mathbf{x}}_t^i \leftarrow \sum_{j=1}^{M} W_{ij}\mathbf{x}_t^j$$

4: *Gradient Estimation*: At each node i, employ KWSA using (4) to estimate gradient $\mathbf{g}_t^i$
5: *Aggregating*: approximate the average gradient:

$$\mathbf{G}_t^j = \bar{\mathbf{g}}_{t-1}^i + \mathbf{g}_t^i - \mathbf{g}_{t-1}^i$$
$$\bar{\mathbf{g}}_t^i \leftarrow \sum_{j=1}^{M} W_{ij}\mathbf{G}_t^j$$

6: *Frank-Wolfe Step*: update

$$\mathbf{x}_{t+1}^i \leftarrow (1-\gamma_t)\bar{\mathbf{x}}_t^i + \gamma_t\mathbf{v}_t^i \text{ where } \mathbf{v}_t^i \in \arg\min_{\mathbf{v}\in\mathcal{C}}\langle\bar{\mathbf{g}}_t^i, \mathbf{v}\rangle,$$

for all agent $i \in [M]$ and $\gamma_t \in (0, 1]$ is a step size.
7: **end for**
8: **Return**: $\bar{\mathbf{x}}_{t+1}^i, \forall\ i \in [M]$.

For notational simplicity and brevity, we employ KWSA for gradient estimation. Similar results can also be obtained for RDSA- and I-RDSA-based gradient approximation schemes. The performance of Algorithm 4 depends on how well the averaged iterates and the averaged gradient estimates are tracked across the network. Before performing the analysis, we state the following assumption regarding the connectivity of the communication graph.

Assumption A6: The interagent communication graph is connected, that is, $\lambda_2(\mathbf{L}) > 0$.

We first establish the bounds concerning the tracking of the averaged iterate and the averaged gradient estimate, for which we use the bounding technique used in [57], where the authors study a distributed first-order FW method. For the averaged iterate tracking we have, for a step size $\gamma_t = 1/t^\alpha$ with $\alpha \in (0, 1]$ in Algorithm 4, $\bar{\mathbf{x}}_t^i$ satisfies

$$\max_{i\in[M]} \|\bar{\mathbf{x}}_t^i - \bar{\mathbf{x}}_t\|_2 = O\left(\frac{1}{t^\alpha}\right) \quad \forall\ t \geq 1 \tag{30}$$

where $\bar{\mathbf{x}}_t$ is the network-averaged iterate at time t. Similarly, we have for the averaged gradient tracking, with

$c_t = \gamma_t/d$ for KWSA

$$\max_{i\in[M]} \|\bar{\mathbf{g}}_t^i - \bar{\mathbf{g}}_t\|_2 = O\left(\frac{1}{t^\alpha}\right) \quad \forall\, t \geq 1 \tag{31}$$

where $\bar{\mathbf{g}}_t$ is the network-averaged gradient estimate at time t evaluated using KWSA. It is to be noted that the error bound with respect to the network average of the iterates and the gradient estimates depends on the connectivity of the graph. Before proceeding to the main results, we mention the key observation which drives the main results of Algorithm 4. The analysis essentially depends on bounding $\left\|\bar{\mathbf{g}}_t^i - \nabla f(\bar{\mathbf{x}}_t)\right\|$. We also note the following equivalence, which plays a key role in order to establish a suitable upper bound for $\left\|\bar{\mathbf{g}}_t^i - \nabla f(\bar{\mathbf{x}}_t)\right\|$:

$$\frac{1}{M}\sum_{i=1}^M \bar{\mathbf{g}}_t^i = \frac{1}{M}\sum_{i=1}^M \mathbf{G}_t^i = \frac{1}{M}\sum_{i=1}^M \mathbf{g}_t^i = \bar{\mathbf{g}}_t$$

where the first equality follows from (29) and the second equality can be shown with a simple induction argument. With the above development, we are now ready to state the convergence results concerning Algorithm 4.

Theorem 5 (Convex Losses): Let Assumptions A1–A6 hold and let each f_i be L-smooth. Let the sequence γ_t be given by $\gamma_t = 2/(t+1)$. In order to achieve a primal suboptimality gap of ϵ for convex loss functions in Algorithm 4, that is,

$$f(\bar{\mathbf{x}}_t) - f(\mathbf{x}^*) \leq \epsilon \tag{32}$$

the number of queries to the oracle is given by $O\left(d/\epsilon\right)$.

Proof: We first note that

$$\bar{\mathbf{x}}_{t+1} = \bar{\mathbf{x}}_t + \gamma_t\left(\sum_{i=1}^M \mathbf{v}_t^i - \bar{\mathbf{x}}_t\right).$$

Define $h_t = f(\bar{\mathbf{x}}_t) - f(\mathbf{x}^*)$. Then, we have by using L-smoothness of f and the finite diameter of the constraint set

$$h_{t+1} \leq h_t + \frac{\gamma_t}{M}\langle \mathbf{v}_t^i - \bar{\mathbf{x}}_t, \nabla f(\bar{\mathbf{x}}_t)\rangle + \frac{\gamma_t^2 LR^2}{2}. \tag{33}$$

We have the following chain of inequalities for each agent i:

$$\begin{aligned}
&\langle \mathbf{v}_t^i - \bar{\mathbf{x}}_t, \nabla f(\bar{\mathbf{x}}_t)\rangle \\
&\quad\leq \langle \mathbf{v}_t^i - \bar{\mathbf{x}}_t, \bar{\mathbf{g}}_t^i\rangle + R\left\|\bar{\mathbf{g}}_t^i - \nabla f(\bar{\mathbf{x}}_t)\right\| \\
&\quad\leq \langle \mathbf{v} - \bar{\mathbf{x}}_t, \bar{\mathbf{g}}_t^i\rangle + R\left\|\bar{\mathbf{g}}_t^i - \nabla f(\bar{\mathbf{x}}_t)\right\| \quad \forall v \in \mathcal{C} \\
&\quad\leq \langle \mathbf{v} - \bar{\mathbf{x}}_t, \nabla f(\bar{\mathbf{x}}_t)\rangle + 2R\left\|\bar{\mathbf{g}}_t^i - \nabla f(\bar{\mathbf{x}}_t)\right\| \quad \forall v \in \mathcal{C}
\end{aligned} \tag{34}$$

where we use the fact that $\mathbf{v}_t^i = \operatorname{argmin}_{\mathbf{v}\in\mathcal{C}}\langle \bar{\mathbf{g}}_t^i, \mathbf{v}\rangle$. Now, we have

$$\begin{aligned}
\left\|\bar{\mathbf{g}}_t^i - \nabla f(\bar{\mathbf{x}}_t)\right\| \leq{}& \left\|\bar{\mathbf{g}}_t - \frac{1}{M}\sum_{i=1}^M \nabla f_i(\bar{\mathbf{x}}_t^i)\right\| + \left\|\bar{\mathbf{g}}_t^i - \bar{\mathbf{g}}_t\right\| \\
&+ \left\|\frac{1}{M}\sum_{i=1}^M \nabla f_i(\bar{\mathbf{x}}_t^i) - \nabla f(\bar{\mathbf{x}}_t)\right\|
\end{aligned} \tag{35}$$

where for the second and third terms, we use the bound derived in (31), where we use $\alpha = 1$. For the first term, we note that for KWSA with $c_t = \gamma_t/d$,

$$\left\|\bar{\mathbf{g}}_t - \frac{1}{M}\sum_{i=1}^M \nabla f_i(\bar{\mathbf{x}}_t^i)\right\| \leq \frac{R\gamma_t}{2}.$$

Using the above-mentioned comparisons, we have that

$$\left\|\bar{\mathbf{g}}_t^i - \nabla f(\bar{\mathbf{x}}_t)\right\| \leq c_1\gamma_t,$$

where c_1 is an appropriately chosen constant depending on the different algorithm parameters, that is, L, R, and M. Substituting back into (33) and defining $\bar{\mathbf{v}}_t \in \operatorname{argmin}_{\mathbf{v}\in\mathcal{C}}\langle \nabla f(\bar{\mathbf{x}}_t), \mathbf{v}\rangle$, we have

$$h_{t+1} \leq h_t + \gamma_t\langle \bar{\mathbf{v}}_t - \bar{\mathbf{x}}_t, \nabla f(\bar{\mathbf{x}}_t)\rangle + c_2\gamma_t^2. \tag{36}$$

Note that

$$\langle \bar{\mathbf{v}}_t - \bar{\mathbf{x}}_t, \nabla f(\bar{\mathbf{x}}_t)\rangle \leq \langle \mathbf{x}^* - \bar{\mathbf{x}}_t, \nabla f(\bar{\mathbf{x}}_t)\rangle \leq -h_t$$

where the first inequality holds for optimality of $\bar{\mathbf{v}}_t$ and the second inequality holds because of convexity of f. Using this inequality in (36) yields a recursion of the form

$$h_{t+1} \leq (1-\gamma_t)h_t + c_2\gamma_t^2$$

and thus we can conclude that $h_t = O(1/t)$ and as KWSA uses d queries per gradient estimate, we have the result as stated. □

Theorem 6 (Nonconvex Losses): Let Assumptions A1–A6 hold and let each f_i be L-smooth and G-Lipschitz. Let the step size be chosen to be $\gamma_t = t^{-1/2}$. Then, in order to obtain a duality gap of ϵ for nonconvex loss functions in Algorithm 4, that is,

$$\min_{t=\lfloor T/2\rfloor+1,\ldots,T} \mathcal{G}(\bar{\mathbf{x}}_t) \leq \epsilon \tag{37}$$

the number of queries to the IZO is given by $O\left(d/\epsilon^2\right)$.

Proof: By the L-smoothness of f, we have

$$f(\bar{\mathbf{x}}_{t+1}) \leq f(\bar{\mathbf{x}}_t) + \langle \nabla f(\bar{\mathbf{x}}_t), \bar{\mathbf{x}}_{t+1} - \bar{\mathbf{x}}_t\rangle + \frac{L}{2}\|\bar{\mathbf{x}}_{t+1} - \bar{\mathbf{x}}_t\|^2$$

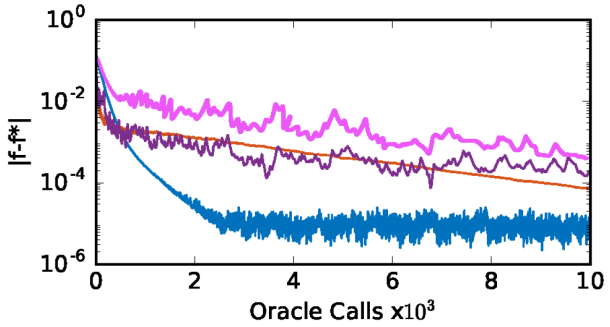

Fig. 1. *Covtype data set: Summary of comparison between first-order PGD (blue), first-order FW (red), decentralized stochastic zeroth-order FW (maroon), and distributed zeroth-order FW (pink).*

and

$$\bar{\mathbf{x}}_{t+1} = \bar{\mathbf{x}}_t + \gamma_t \left(\sum_{i=1}^{M} \mathbf{v}_t^i - \bar{\mathbf{x}}_t \right).$$

As $\mathbf{v}_t^i, \bar{\mathbf{x}}_t \in \mathcal{C}$, we have that $\|\bar{\mathbf{x}}_{t+1} - \bar{\mathbf{x}}_t\| \leq R\gamma_t$. Using (34) and (35) for $\alpha = 1/2$, we have that

$$\begin{aligned} f(\bar{\mathbf{x}}_{t+1}) &\leq f(\bar{\mathbf{x}}_t) - \gamma_t \langle \nabla f(\bar{\mathbf{x}}_t), \bar{\mathbf{x}}_t - \bar{\mathbf{v}}_t \rangle + c_3 \gamma_t^2 \\ &\leq f(\bar{\mathbf{x}}_t) - \gamma_t \mathcal{G}(\bar{\mathbf{x}}_t) + c_3 \gamma_t^2 \end{aligned}$$

where we used the fact that $\mathcal{G}(\bar{\mathbf{x}}_t) = \max_{\mathbf{v}\in\mathcal{C}} \langle \nabla f(\bar{\mathbf{x}}_t), \bar{\mathbf{x}}_t - \mathbf{v} \rangle = \max_{\mathbf{v}\in\mathcal{C}} \langle \nabla f(\bar{\mathbf{x}}_t), \bar{\mathbf{x}}_t - \bar{\mathbf{v}}_t \rangle$ and c_3 is an appropriately chosen constant which depends on the algorithm parameters. Rearranging and summing from $t = \lfloor T/2 \rfloor + 1$ to $t = T$, we have that

$$\begin{aligned} \sum_{t=\lfloor T/2 \rfloor+1}^{T} \gamma_t \mathcal{G}(\bar{\mathbf{x}}_t) &\leq f(\bar{\mathbf{x}}_{\lfloor T/2 \rfloor+1}) - f(\bar{\mathbf{x}}_T) + \sum_{t=\lfloor T/2 \rfloor+1}^{T} c_3 \gamma_t^2 \\ &\leq GR + c_3(1 + \log 2) = GR + c_4. \end{aligned}$$

We also have that

$$\begin{aligned} \sum_{t=\lfloor T/2 \rfloor+1}^{T} \gamma_t \mathcal{G}(\bar{\mathbf{x}}_t) &\geq \min_{t=\lfloor T/2 \rfloor+1,\ldots,T} \mathcal{G}(\bar{\mathbf{x}}_t) \sum_{t=\lfloor T/2 \rfloor+1}^{T} \gamma_t \\ &\geq \min_{t=\lfloor T/2 \rfloor+1,\ldots,T} \mathcal{G}(\bar{\mathbf{x}}_t)\, c_5 \sqrt{T}. \end{aligned}$$

Combining the above inequalities yields

$$\min_{t=\lfloor T/2 \rfloor+1,\ldots,T} \mathcal{G}(\bar{\mathbf{x}}_t) = O\left(\frac{1}{\sqrt{T}}\right).$$

Noting that KWSA requires d queries for each gradient estimate, we have the result as stated. □

It is worth noting that the problem being considered in this section is a deterministic version of FW and hence a better query complexity is obtained in comparison to stochastic FW methods. The dependence of network connectivity is not explicit in the query complexity, but instead is implicit in the tracking capacity of the network in terms of the averaged iterate and the averaged gradient estimates. A sparsely connected network would lead to slower network tracking, while a densely connected network would result in faster tracking. The query complexity in our results is in the order sense and specifically explicitly quantify the dimension dependence and the iteration dependence.

VIII. APPLICATIONS AND EXPERIMENTS

A. Convex Loss Functions

In this section, to study the performance of constrained stochastic optimization and to understand the dependence of the primal gap in terms of oracle calls, we solve a simple lasso regression on the data set covtype ($n = 581012$, $d = 54$) from LIBSVM website.[2] We use the variant with feature values in $[0, 1]$ and solve the following problem:

$$\min_{\|\mathbf{w}\|_1 \leq 1} \quad \frac{1}{2} \|\mathbf{y} - \mathbf{X}^\top \mathbf{w}\|_2^2$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$ represents the feature vectors and $\mathbf{y} \in \mathbb{R}^n$ are the corresponding targets. To benchmark the performance of the proposed algorithm, we compare them with first-order PGD and first-order FW. For the decentralized and distributed setting, we consider 20 nodes and equally divide the data points among all the nodes. Furthermore, for the distributed setting, we consider a 20-node graph with $\|\mathbf{W} - \mathbf{J}\| = 0.36$, where $\mathbf{W}$ is the weight matrix of the network under consideration and $\mathbf{J} = \mathbf{1}\mathbf{1}^\top/20$ corresponds to that of a completely connected graph. The metric $\|\mathbf{W} - \mathbf{J}\|$ closely replicates the connectivity and the information flow in the network. While $\|\mathbf{W} - \mathbf{J}\| = 0$ corresponds to a completely connected graph, $\|\mathbf{W} - \mathbf{J}\| = 1$ corresponds to the case when the nodes do not collaborate with each other. All the FW-based algorithms pay a price for the projection-free nature of the algorithm and are as depicted in Fig. 1. However, in spite of being zeroth-order algorithms and the associated problem dimension being $d = 54$, the distributed and decentralized zeroth-order FW algorithms perform almost as good as their first-order counterpart. For the zeroth-order FW algorithms, we used I-RDSA with $m = 6$. While the performance of the decentralized zeroth-order stochastic FW matches that of first-order FW, the performance of distributed zeroth-order FW is fairly close to its decentralized counterpart. This showcases zeroth-order FW-based methods to be effective choices for moderate dimensional ($d < 100$) constrained optimization problems.

[2]Available at https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

B. Nonconvex Loss Functions

In this section, we focus on adversarial attacks on machine learning models where black-box optimization methods have been successfully used recently. Adversarial attacks are an important tool to analyze and certify robustness of a deployable machine learning model. We give a brief overview of adversarial attacks before proceeding further.

In the context of classification for a classifier, adversarial examples are carefully designed inputs to the model, that is, the classifier which have been perturbed or distorted so as to make the output of the model erroneous. For example, an adversarial example would be a perturbed image of a pig which would still be a pig to a human eye, while the classifier would now output the class to be that of an airplane. While any perturbation can be made to misclassify an input image, it is important to ensure that there is minimal visual distortion to a human eye. Typically, the visual distortion is prespecified in terms of ℓ_p-norm, for some fixed p, less than some ϵ_p. Furthermore, in the context of classifiers, adversarial attacks can be further classified into untargeted and targeted attacks. By untargeted attacks, we mean attack strategies which are aimed at only misclassifying the input image to any other class except the true class. However, targeted attacks are focused at misclassifying the input image to a target class. For the rest of the discussion, we focus on untargeted attacks.

Formally, a classifier may be defined as $C : \mathcal{X} \mapsto \mathcal{Y}$ with a corresponding loss function $L(\mathbf{x}, y)$ for classification, where $\mathbf{x} \in \mathcal{X}$ represents the input to the classifier, $y \in \mathcal{Y}$ represents the corresponding true class of the input, while $\mathcal{X}$ is the set of inputs and $\mathcal{Y}$ is the set of labels. Technically speaking, the procedure of generating a misclassified example can be posed as an optimization problem. The procedure to generate an adversarial example can be written down as finding $\mathbf{x}'$, for a given input $\mathbf{x}$ which maximizes $L(\mathbf{x}', y)$ while still adhering to the ϵ_p-closeness in terms of a specified metric, to the original input. Formally, the aforementioned constrained optimization can be cast in the following way:

$$\mathbf{x}_{\text{adv}} = \underset{\mathbf{x}':\|\mathbf{x}'-\mathbf{x}\|_p \leq \epsilon_p}{\operatorname{argmax}} L(\mathbf{x}', y). \tag{38}$$

Furthermore, adversarial attacks can be categorized into white-box and black-box attacks based on the information accessible to the attacker. White-box settings consist of scenarios where the entire classifier and the trained parameter values along with the analytical form of the classifier loss function is available to the attacker. Several white-box attack techniques such as fast gradient signed method (FGSM) (see [17] for a detailed treatment), PGD has been particularly effective in generating adversarial examples for state-of-the-art deep networks built for challenging computer vision data sets. While FGSM can only handle ℓ_∞ constraints when it comes to visual distortion, PGD can handle all sorts of ℓ_p constraints.

However, in most real-world deployments, it is impractical to assume complete access to the classifier and analytic form of the corresponding loss function, which makes black-box settings more realistic.

In a typical black-box setting, the adversary has access only to a zeroth-order oracle, which when queried for an input $(\mathbf{x}, y)$ yields the value of the loss function $L(\mathbf{x}, y)$. In spite of the information constraints and typically high-dimensional inputs, black-box attacks have been shown to be pretty effective [22], [23], [40]. Black-box adversarial attacks can be broadly categorized across a few different dimensions: optimization-based versus transfer-based attacks, and score-based versus decision-based attacks.

In the optimization-based adversarial setting, the adversarial attack is formulated as a maximization of the classification loss function which can be the accuracy of the classifier or some continuous variant of it while accessing a zeroth-order oracle. The zeroth-order information can be further categorized into score-based and decision-based attacks. By score-based attacks, we mean scenarios in which the attacker directly observes a model loss, class probability, or some other continuous output of the classifier on a given example, whereas in decision-based attacks, the attacker gets to observe only the hard label predicted by the classifier. Decision-based attacks have been studied in [3], [5], and [6] and typically require more queries to the classifier than the score-based setting.

In the regime of score-based attacks, the first such iterative attack on a class of binary classifiers was studied in [45]. A real-world application of black-box attacks to fool a PDF malware classifier was demonstrated by Xu *et al.* [61], for which a genetic algorithm was used. This article [42] demonstrated the first black-box attack on deep neural networks. Subsequently, black-box attacks based on zeroth-order optimization schemes, using techniques such as KWSA and RDSA, were developed in [6] and [22]. Though Chen *et al.* [6] generated successful attacks attaining high attack accuracy, the method was found to be extremely query hungry which was then remedied to an extent by Ilyas *et al.* [22]. Ilyas *et al.* [23] exploited correlation of gradients across iterations by setting a prior and use a piecewise constant perturbation, that is, tiling to develop a query efficient black-box method. In addition to that, the method developed in [23] is essentially a stochastic FW-based method for the ℓ_2 constraint-based adversarial attack. However, all the aforementioned attacks generate perturbations for each image separately, which brings us to the question, whether one perturbation can be generated which misclassifies many images and not just one? This leads us to the problem of finding universal adversarial perturbation [41] which can be formalized as follows as a constrained stochastic optimization problem:

$$\boldsymbol{\delta}' = \underset{\|\boldsymbol{\delta}\|_p \leq \epsilon_p}{\operatorname{argmax}} \mathbb{E}_{(\mathbf{x},y)\in(\mathcal{X},\mathcal{Y})} \left[L(\mathbf{x} + \boldsymbol{\delta}, y) \right]. \tag{39}$$

Given the need to take a pass over possibly high-dimensional data for the optimization problem at hand

in (39) and the huge dimension and size of computer vision data sets such as MNIST [30] where each image is 784-dimensional, this problem turns out to be a perfect real-world application for decentralized constrained stochastic optimization and as such for decentralized stochastic FW which we demonstrate below. A crucial point to note here is that it is not necessary to find the global maximizer in (39). Instead, it is just enough to find a "good enough" feasible point in the constraint set which misclassifies most images in the set. While the MNIST data set in itself is not prohibitively large, other widely used data sets for computer vision tasks such as ImageNet [11] are prohibitively large to be processed by a single machine. Universal perturbations are particularly useful for adversarial training of a neural network, which typically require multiple universal perturbations to effectively defend against adversaries. As universal perturbations involve an expensive optimization problem, they can be particularly hard to find. A distributed implementation, where each agent generates its own universal perturbation, provides diversified perturbations which make it highly relevant for adversarial training.

We perform an experiment on MNIST to find a universal adversarial perturbation. We use the pretrained LeNet-5 model available from Pytorch to demonstrate the attacks. We enforce the information access limited to the softmax layer, that is, only score function values can be obtained for different classes after feeding an input image. We set the constraint to be that of ℓ_∞ with $\epsilon_p = 0.25$ in (39). We use all the correctly classified images from the 10 000 images (scaled to [0, 1]) in the MNIST test set. We use the decentralized stochastic FW algorithm to solve this problem and benchmark it against the centralized version. We pick 100 images from each class and estimate gradients for each image using 20, 50, and 100 queries, respectively. The attack accuracy is then reported by adding the perturbation to the correctly classified images from the 10 000 images (scaled to [0, 1]) in the MNIST test set. For the decentralized setting, we consider a set of ten worker nodes with 100 images each, with ten images drawn from each class. For the distributed setting, we also consider a network of ten nodes with 100 images each, with ten images drawn from each class. The network is reasonably well connected with $\|\mathbf{W} - \mathbf{J}\| = 0.43$, where $\mathbf{W}$ is the weight matrix of the network under consideration and $\mathbf{J} = \mathbf{1}\mathbf{1}^\top/10$ corresponds to that of a completely connected graph. In order to generate the universal perturbation, we average the δ's obtained at each of the nodes for the distributed setting. Figs. 2–4 provide some examples of the universal adversarial perturbation generated. Table 1 summarizes the performance of a distributed stochastic FW and benchmarks it with respect to the decentralized FW. As it can be seen from Table 1, the performance of the distributed scheme in terms of the attack success rate is very close to that of its decentralized counterpart.

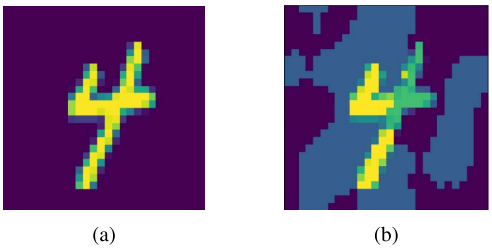

Fig. 2. *Image of 4 changed to 8 with the adversarial perturbation. (a) Original image: Class 4. (b) Perturbed image: Class 8.*

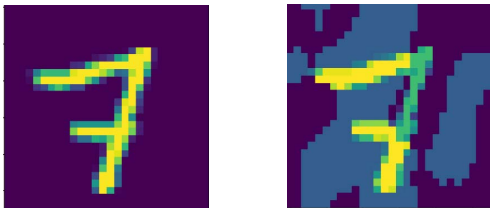

Fig. 3. *Image of 7 changed to 3 with the adversarial perturbation. (a) Original image: Class 7. (b) Perturbed image: Class 3.*

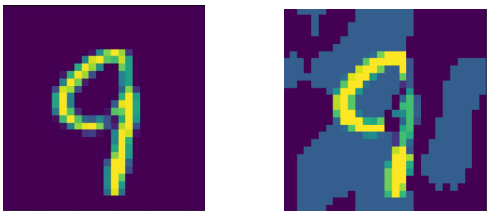

Fig. 4. *Image of 9 changed to 8 with the adversarial perturbation. (a) Original image: Class 9. (b) Perturbed image: Class 8.*

Table 1 Summary of ℓ_∞ Universal Adversarial Perturbation With $\epsilon = 0.25$ MNIST Attacks Using Decentralized and Distributed FW. The Number of Queries in the Columns Denote the Number of Queries Used Per Image

Attack	20 queries	50 queries	100 queries
Decentralized FW	30.08%	41.38%	57.73%
Distributed FW	26.21%	38.62%	51.42%

IX. CONCLUSION

In this article, we have focused on the problem of constrained stochastic optimization built around zeroth-order oracles. In particular, we have reviewed latest developments in zeroth-order FW frameworks for decentralized and distributed constrained optimization. We have illustrated quantitative aspects of the performance of the approaches in terms of number of queries and also in terms of number of iterations. Finally, we have discussed the effectiveness and the applicability of stochastic FW frameworks for black-box adversarial attacks in the context of deep neural networks. ■

REFERENCES

[1] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, "Stochastic gradient push for distributed deep learning," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 344–353.

[2] K. Balasubramanian and S. Ghadimi, "Zeroth-order (non)-convex stochastic optimization via conditional gradient and gradient updates," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3459–3468.

[3] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," 2017, *arXiv:1712.04248*. [Online]. Available: http://arxiv.org/abs/1712.04248

[4] S. Bubeck, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Found. Trends Mach. Learn.*, vol. 5, no. 1, pp. 1–122, 2012.

[5] J. Chen, M. I. Jordan, and M. J. Wainwright, "HopSkipJumpAttack: A query-efficient decision-based attack," 2019, *arXiv:1904.02144*. [Online]. Available: http://arxiv.org/abs/1904.02144

[6] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, New York, NY, USA, 2017, pp. 15–26.

[7] X. Chen *et al.*, "Zo-AdaMM: Zeroth-order adaptive momentum method for black-box optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 7202–7213.

[8] K. Choromanski, M. Rowland, V. Sindhwani, R. E. Turner, and A. Weller, "Structured evolution with compact architectures for scalable policy optimization," 2018, *arXiv:1804.02395*. [Online]. Available: http://arxiv.org/abs/1804.02395

[9] F. R. K. Chung, *Spectral Graph Theory*. Providence, RI, USA: AMS, 1997.

[10] S. N. Deming, L. R. Parker, Jr., and M. B. Denton, "A review of simplex optimization in analytical chemistry," *CRC Crit. Rev. Anal. Chem.*, vol. 7, no. 3, pp. 187–202, 1978.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[12] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono, "Optimal rates for zero-order convex optimization: The power of two function evaluations," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2788–2806, May 2015.

[13] M. Eisen, A. Mokhtari, and A. Ribeiro, "Decentralized quasi-Newton methods," *IEEE Trans. Signal Process.*, vol. 65, no. 10, pp. 2613–2628, May 2017.

[14] C. Fang, C. J. Li, Z. Lin, and T. Zhang, "SPIDER: Near-optimal non-convex optimization via stochastic path-integrated differential estimator," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 689–699.

[15] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Res. Logistics Quart.*, vol. 3, nos. 1–2, pp. 95–110, 1956.

[16] D. Garber and E. Hazan, "Faster rates for the Frank-Wolfe method over strongly-convex sets," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 541–549.

[17] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*. [Online]. Available: http://arxiv.org/abs/1412.6572

[18] G. A. Gray, T. G. Kolda, K. Sale, and M. M. Young, "Optimizing an empirical scoring function for transmembrane protein structure determination," *INFORMS J. Comput.*, vol. 16, no. 4, pp. 406–418, Nov. 2004.

[19] D. Hajinezhad, M. Hong, and A. Garcia, "Zeroth order nonconvex multi-agent optimization over networks," 2017, *arXiv:1710.09997*. [Online]. Available: http://arxiv.org/abs/1710.09997

[20] E. Hazan and S. Kale, "Projection-free online learning," in *Proc. 29th Int. Conf. Mach. Learn.*, 2012, pp. 1843–1850.

[21] E. Hazan and H. Luo, "Variance-reduced and projection-free stochastic optimization," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 1263–1271.

[22] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 2137–2146.

[23] A. Ilyas, L. Engstrom, and A. Madry, "Prior convictions: Black-box adversarial attacks with bandits and priors," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019.

[24] M. Jaggi, "Revisiting Frank-Wolfe: Projection-free sparse convex optimization," in *Proc. 30th Int. Conf. Mach. Learn.*, Atlanta, GA, USA, Jun. 2013, pp. 427–435.

[25] K. Ji, Z. Wang, Y. Zhou, and Y. Liang, "Improved zeroth-order variance reduced algorithms and analysis for nonconvex optimization," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 3100–3109.

[26] S. Kar, J. M. F. Moura, and K. Ramanan, "Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect communication," *IEEE Trans. Inf. Theory*, vol. 58, no. 6, pp. 3575–3605, Jun. 2012.

[27] J. Kiefer and J. Wolfowitz, "Stochastic estimation of the maximum of a regression function," *Ann. Math. Statist.*, vol. 23, no. 3, pp. 462–466, Sep. 1952.

[28] S. Lacoste-Julien, "Convergence rate of Frank-Wolfe for non-convex objectives," 2016, *arXiv:1607.00345*. [Online]. Available: http://arxiv.org/abs/1607.00345

[29] S. Lacoste-Julien and M. Jaggi, "On the global linear convergence of Frank-Wolfe optimization variants," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 496–504.

[30] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[31] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5330–5340.

[32] S. Liu, J. Chen, P.-Y. Chen, and A. Hero, "Zeroth-order online alternating direction method of multipliers: Convergence analysis and applications," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2018, pp. 288–297.

[33] S. Liu, B. Kailkhura, P.-Y. Chen, P. Ting, S. Chang, and L. Amini, "Zeroth-order stochastic variance reduction for nonconvex optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3727–3737.

[34] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3122–3136, Jul. 2008.

[35] D. Malik, A. Pananjady, K. Bhatia, K. Khamaru, P. Bartlett, and M. Wainwright, "Derivative-free methods for policy optimization: Guarantees for linear quadratic systems," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 2916–2925.

[36] A. L. Marsden, J. A. Feinstein, and C. A. Taylor, "A computational framework for derivative-free optimization of cardiovascular geometries," *Comput. Methods Appl. Mech. Eng.*, vol. 197, nos. 21–24, pp. 1890–1905, Apr. 2008.

[37] A. L. Marsden, M. Wang, J. E. Dennis, and P. Moin, "Trailing-edge noise reduction using derivative-free optimization and large-eddy simulation," *J. Fluid Mech.*, vol. 572, pp. 13–36, Feb. 2007.

[38] A. Mokhtari, H. Hassani, and A. Karbasi, "Conditional gradient method for stochastic submodular maximization: Closing the gap," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2018, pp. 1886–1895.

[39] A. Mokhtari, Q. Ling, and A. Ribeiro, "An approximate Newton method for distributed optimization," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 2959–2963.

[40] S. Moon, G. An, and H. O. Song, "Parsimonious black-box adversarial attacks via efficient combinatorial optimization," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 4636–4645.

[41] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1765–1773.

[42] N. Narodytska and S. Kasiviswanathan, "Simple black-box adversarial attacks on deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1310–1318.

[43] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[44] A. Nedic and A. Olshevsky, "Stochastic gradient-push for strongly convex functions on time-varying directed graphs," *IEEE Trans. Autom. Control*, vol. 61, no. 12, pp. 3936–3947, Dec. 2016.

[45] B. Nelson *et al.*, "Query strategies for evading convex-inducing classifiers," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 1293–1332, 2012.

[46] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Norwell, MA, USA: Kluwer, 2004.

[47] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Found. Comput. Math.*, vol. 17, no. 2, pp. 527–566, 2017.

[48] S. J. Reddi, S. Sra, B. Póczos, and A. Smola, "Stochastic Frank-Wolfe methods for nonconvex optimization," in *Proc. 54th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2016, pp. 1244–1251.

[49] A. Ruszczyński, "A merit function approach to the subgradient method with averaging," *Optim. Methods Softw.*, vol. 23, no. 1, pp. 161–172, Feb. 2008.

[50] A. K. Sahu, D. Jakovetic, D. Bajovic, and S. Kar, "Distributed zeroth order optimization over random networks: A Kiefer-Wolfowitz stochastic approximation approach," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2018, pp. 4951–4958.

[51] A. K. Sahu and S. Kar. (2020). *Decentralized Zeroth Order Constrained Stochastic Optimization Algorithms: Frank-Wolfe and Variants With Applications to Black-Box Adversarial Attacks*. [Online]. Available: https://anitksahu.github.io/pieee.pdf

[52] A. K. Sahu, M. Zaheer, and S. Kar, "Towards gradient free and projection free stochastic optimization," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 3468–3477.

[53] Z. Shen, C. Fang, P. Zhao, J. Huang, and H. Qian, "Complexities in projection-free stochastic non-convex minimization," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 2868–2876.

[54] Y. Tang and N. Li, "Distributed zero-order algorithms for nonconvex multi-agent optimization," in *Proc. 57th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2019, pp. 781–786.

[55] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Trans. Autom. Control*, vol. AC-31, no. 9, pp. 803–812, Sep. 1986.

[56] R. Tutunov, H. Bou-Ammar, and A. Jadbabaie, "Distributed Newton method for large-scale consensus optimization," *IEEE Trans. Autom. Control*, vol. 64, no. 10, pp. 3983–3994, Oct. 2019.

[57] H.-T. Wai, J. Lafond, A. Scaglione, and E. Moulines, "Decentralized Frank–Wolfe algorithm for convex and nonconvex problems," *IEEE Trans. Autom. Control*, vol. 62, no. 11, pp. 5522–5537, Nov. 2017.

[58] S. Wang, F. Roosta, P. Xu, and M. W. Mahoney, "GIANT: Globally improved approximate Newton method for distributed optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 2332–2342.

[59] Y. Wang, S. Du, S. Balakrishnan, and A. Singh, "Stochastic zeroth-order optimization in high dimensions," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2018, pp. 1356–1365.
[60] R. Xin, A. K. Sahu, U. A. Khan, and S. Kar, "Distributed stochastic optimization with gradient tracking over strongly-connected networks," in *Proc. IEEE 58th Conf. Decis. Control (CDC)*, Dec. 2019, pp. 8353–8358.
[61] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers," in *Proc. Netw. Distrib. Syst. Symp.*, 2016, pp. 21–24.
[62] A. Yurtsever, S. Sra, and V. Cevher, "Conditional gradient methods via stochastic path-integrated differential estimator," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019 pp. 7282–7291.

ABOUT THE AUTHORS

Anit Kumar Sahu (Member, IEEE) received the B.Tech. degree in electronics and electrical communication engineering and the M.Tech. degree in telecommunication systems engineering from IIT Kharagpur, Kharagpur, India, in May 2013, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in December 2018.

Since January 2019, he has been with the Bosch Center for Artificial Intelligence, Pittsburgh, as a Machine Learning Research Scientist. His research interests include stochastic optimization, robust machine learning, and distributed inference in large-scale systems.

Dr. Sahu is a recipient of the A. G. Jordan Award from the Department of Electrical and Computer Engineering, Carnegie Mellon University.

Soummya Kar (Member, IEEE) received the B.Tech. degree in electronics and electrical communication engineering from IIT Kharagpur, Kharagpur, India, in May 2005, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2010.

From June 2010 to May 2011, he was with the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA, as a Postdoctoral Research Associate. He is currently an Associate Professor of electrical and computer engineering with Carnegie Mellon University. His research interests include decision-making in large-scale networked systems, stochastic systems, multiagent systems and data science, and with applications in cyber–physical systems and smart energy systems.