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

Anit Kumar Sahu and Soummya Kar

***Abstract*—Zeroth order optimization algorithms are an attractive alternative for stochastic optimization problems, when gradient computations are expensive or when closed form for 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 stochastic optimization, specifically focusing on constrained optimization settings. Firstly, we describe convergence rates and the dimension dependence for a distributed projection free algorithm catered to solve constrained stochastic optimization problems. Secondly, we describe and quantify convergence rates for a variance reduced distributed zeroth order optimization method inspired from 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.**

1. Introduction

The growth of machine learning and data-driven methods over the past few years has been largely 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. However, 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 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], [16], [16], [34], [35]. 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 [2], [4], [5], [21]. 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 associated machine learning models may be readily *attacked* to misclassify [15]. Designing adversarial attacks, for the purpose of validating and strengthening models, has seen growing interest in the machine learning community [2], [4], [5], [21]. Black-box optimization methods and in particular black-box gradient estimation techniques have been particularly effective in black-box attacks [6] on neural networks, i.e., 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 [33], and optimization with bandit feedback [3].

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 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 which are able to operate without expensive gradient computations and are able to function with 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 paper.

In this paper, we focus on constrained stochastic optimization problems where the optimizer has access to (unbiased) loss function evaluations, i.e., a zeroth order oracle [42]. In a constrained optimization setting with access to a zeroth order oracle, it is natural to develop and employ analogues 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-

Anit Kumar Sahu is with the Bosch Center for Artificial Intelligence, Pittsburgh, PA 15222. anit.sahu@gmail.com

Soummya Kar is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 soummyak@andrew.cmu.edu

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

order analogue of PGD could suffer from slow convergence and poor dependence on the problem parameters (such as dimension). This paper focuses on Frank-Wolfe type algorithms [22] and their distributed counterparts, intuitively, because of the flexibility available to avoid projection while only requiring approximate minimization of the linear minimization oracle encountered. On the other hand, a zeroth order analogue of Frank-Wolfe avoids a projection operation and the linear minimization encountered in it can be solved up to a degree of inexactness, thereby making it computationally favorable to use. Moreover, in applications such as black-box adversarial attacks which are typically posed as constrained optimization problems, a "good enough" feasible point satisfying the misclassification objective is usually sought for rather than the optimal point (see Section 8 for a relevant illustration). Hence, we focus on *projection free* and *gradient free* constrained stochastic optimization built around the Frank-Wolfe framework.

In this paper, we discuss both basic and variance reduced zeroth order stochastic Frank-Wolfe 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 datacenter 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 fashion, by means of local computation and message exchanges with *neighboring* devices [24], [32], [40], [50], which is typical in IoT type setups.

We end this section by providing a brief discussion of the main variants of zeroth order methods to be reviewed in the paper, in the decentralized and/or distributed setting. In this article, we first present a setting of the vanilla stochastic Frank-Wolfe [1], [47] 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 Frank-Wolfe to circumvent the potential divergence issue due to non-decaying gradient noise and bias, we present a gradient averaging technique used in [36], [45], [47] to get a surrogate gradient estimate which reduces the noise and the associated bias. Intuitively, the gradient averaging technique reduces the linear minimization step in the Frank-Wolfe 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 Frank-Wolfe, we demonstrate techniques based on SPIDER [12]. We then present a distributed version of the deterministic Frank-Wolfe algorithm. where a gradient tracking technique is employed. We present rates for both convex and non-convex loss functions, specifically quantifying the dependence of the primal gap and the Frank-Wolfe 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.

2. Related Work

For constrained stochastic optimization problems, the go to algorithm for most practitioners is the projected gradient descent or PGD (see [42] for a detailed treatment) and further stochastic versions of it so as to avoid the entire gradient computation at one go. However, more often that not, PGD involves a very expensive projection operation which needs to be solved to a reasonable degree of accuracy or else the algorithm gets tends to get stuck. Hence, projection free methods have seen a major surge recently. In the context of projection free methods, [13] proposed the Frank-Wolfe algorithm for smooth convex functions with line search which was then extended to accommodate inexact linear minimization steps without sacrificing convergence performance in [22]. The convergence rates for deterministic Frank-Wolfe, i.e., the gradient information is exact, has been subsequently improved with additional assumptions in [14], [27]. Stochastic versions of Frank-Wolfe 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 [18], [19] so as to obtain faster rates, while [36] studied the version with a mini-batch size of 1. As far as dealing with non-convex losses is concerned, a deterministic Frank-Wolfe algorithm was proposed in [26], while [44], [48], [54] used variance reduction techniques on a stochastic version of Frank-Wolfe and further improved the rates.

Algorithms for zeroth order convex and non-convex optimization with access to zeroth order oracles have been studied in [1], [7], [11], [12], [23], [29], [31], [46], [47], [49], [52], where the oracles considered are either incremental zeroth order oracles (IZO) which returns the exact loss function value at the queried point or stochastic zeroth order oracles (SZO) which return an unbiased estimate of the loss function value at the queried point. In particular, [11] 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 [12], [23], [31]. However, most of the aforementioned works consider unconstrained optimization problems. In regards to constrained zeroth order optimization, [1], [29], [47] studied the problem for both convex and non-convex loss functions. In [29], a projection step was considered so as to cater to the constrained problem, while in [47] a gradient smoothing technique was utilized in addition to a stochastic zeroth-order Frank-Wolfe framework so as to avoid potentially expensive projection operations. The authors in [1] use a proximal mapping to further accelerate the convergence of Frank-Wolfe 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 incremental zeroth order oracle has been studied in [17], [49], while for a stochastic zeroth oracle was studied in [46]. However, distributed zeroth order constrained optimization is relatively

less explored, which is something we discuss in this paper. In this paper, 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 a SZO and we 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 zeroth order schemes it is crucial to quantify the dependence on the dimension of the problem at hand. In this paper, 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.

Paper Organization. Section 4 discusses the preliminaries concerning the Frank-Wolfe algorithm, black-box gradient estimation and stochastic Frank-Wolfe. Algorithms for decentralized stochastic zeroth order Frank-Wolfe algorithm and its convergence results are presented in Section 5. Section 6 presents variance-reduction based algorithms for decentralized stochastic zeroth order Frank-Wolfe algorithm, while in Section 7 algorithms for distributed zeroth order Frank-Wolfe algorithm are discussed and presented. Finally, experimental results are presented in Section 8 and Section 9 concludes the paper.

3. 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 $\mathcal{C} \in \mathbb{R}^d$ 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 apriori). 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 $f(\mathbf{x})$ can not be queried directly. Thus, be it gradients obtained from a stochastic first order oracle (SFO) or function values obtained from a stochastic zeroth order oracle (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, i.e., a scenario in which the data is distributed across M worker nodes, the problem is then posed as finite-sum problem where $\mathcal{P}$ is taken to be a uniform distribution over 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 i-th 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, i.e., the finite sum optimization problem in (28) 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 the paper, 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. Differently 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 [25]) scheme approximates the gradient by sampling the objective function along the canonical basis vectors. Formally, gradient estimate can be expressed as:

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

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, for example [11], [43]). The random directions gradient estimator (RDSA) 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) turn out 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 projected gradient descent (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 also can be potentially expensive leading to the need of looking beyond PGD for zeroth order constrained optimization. The theoretical properties of zeroth-order PGD except for specific constraint sets remains largely unexplored in literature. In particular, it is not clear as to what are the convergence rates are for zeroth order PGD and its dependence on the number of iterations and the dimension of the problem at hand. Moreover, in applications we consider in this paper, i.e., black-box adversarial attacks, as it will become clear later in Section 8, it is important to find a feasible point efficiently which satisfies the constraint than to find the 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 Frank-Wolfe algorithm and its variants in decentralized and distributed settings in this paper.

4. Frank-Wolfe: From first order to zeroth order

We revisit preliminaries pertaining to the classical Frank-Wolfe algorithm next.

A. Background: Frank-Wolfe Algorithm

Frank-Wolfe algorithm involves approximating the objective by a first-order Taylor approximation. In the case, when exact first order information is available, i.e., one has access to a first order oracle, a deterministic Frank-Wolfe method involves the following steps:

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

where $\gamma_t = \frac{2}{t+2}$. 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. In particular, a lot of structured sets (see Table 1 of [22]) admit to low computational complexity based 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,

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

Algorithm 1 Deterministic Zeroth Order Frank Wolfe

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**

where C_1 is a positive constant and the algorithm can be shown to retain the same convergence rate (see, for example [22]). The room for inexactness for the minimization in (6) and it admitting to computationally efficient solutions for widely used structured constraint sets makes Frank-Wolfe an attractive avenue for the class of optimization problems being considered in this paper.

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

B. Deterministic Zeroth Order Frank-Wolfe

The deterministic version of the optimization in (1) can be re-stated as follows:

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

We state the main assumptions that we will be making use of throughout the paper before proceeding further:

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

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

Assumption A3. The function f is L-smooth, i.e., its gradient ∇f is L-Lipschitz continuous over the set $\mathcal{C}$, i.e., for all $x, y \in \mathcal{C}$

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

Assumption A4. The $\mathbf{z}_t$'s are drawn from a distribution μ such that $M(\mu) = \mathbb{E}\left[\left\|\mathbf{z}_t\right\|^6\right]$ 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,

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

Assumption A5. The unbiased gradient estimates, $\nabla F_i\left(\mathbf{x};\mathbf{y}\right)$ of $\nabla f_i(\mathbf{x})$, i.e., $\mathbb{E}_{\mathbf{y}\sim\mathcal{P}_i}\left[\nabla F_i\left(\mathbf{x};\mathbf{y}\right)\right] = \nabla f_i(\mathbf{x})$, $\forall i = 1, \cdots, M$ satisfy

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

In what follows, we demonstrate the equivalence of a typical zeroth order Frank-Wolfe framework and that of an inexact classical Frank-Wolfe optimization. We consider Kiefer-Wolfowitz stochastic approximation (KWSA) for gradient

estimation for this purpose. In particular, the KWSA gradient estimator in (4) can be expressed as follows:

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

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:

$$\langle \mathbf{v}, \mathbf{g}(\mathbf{x}_t)\rangle = \langle \mathbf{v}, \nabla F(\mathbf{x}_t)\rangle$$
$$+ \frac{c_t}{2}\sum_{i=1}^{d}\langle \mathbf{e}_i, \nabla^2 F(\mathbf{x}_t + \lambda c_t \mathbf{e}_i)\mathbf{e}_i\rangle\langle \mathbf{v}, \mathbf{e}_i\rangle$$
$$\Rightarrow \min_{\mathbf{v}\in\mathcal{C}}\langle \mathbf{v}, \mathbf{g}(\mathbf{x}_t)\rangle \leq \min_{\mathbf{s}\in\mathcal{C}}\langle \mathbf{s}, \nabla F(\mathbf{x}_t)\rangle + \frac{c_t L R d}{2}, \quad (11)$$

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

Theorem 4.1 ([47]). *Assume the function $F(\cdot)$ is convex with its gradient ∇F being L-Lipschitz continuous over the set $\mathcal{C}$, i.e., for all $x, y \in \mathcal{C}$*

$$\|\nabla F(\mathbf{x}) - \nabla F(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|. \quad (12)$$

Given the zeroth order Frank-Wolfe algorithm in Algorithm 1, we obtain the following bound:

$$F(\mathbf{x}_t) - F(\mathbf{x}^*) = \frac{Q_{ns}}{t+2}, \quad (13)$$

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

In summary, Theorem 4.1 shows the equivalence of the deterministic zeroth order Frank-Wolfe algorithm and that of the inexact classical Frank-Wolfe 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 Frank-Wolfe algorithm.

C. Stochastic Frank Wolfe

In the classical Frank-Wolfe algorithm, the replacement of the true gradient $\nabla f(\mathbf{x}_k)$ by its stochastic counterpart, i.e., $\nabla F(\mathbf{x}_k; \mathbf{y}_k)$ could make the algorithm divergent due to an additional variance term due to the gradient approximations. In addition to the potential divergence, with 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, for example [47]) to counter this problem which is as follows:

$$\mathbf{d}_t = (1-\rho_t)\mathbf{d}_{t-1} + \rho_t \mathbf{g}(\mathbf{x}_t, \mathbf{y}_t), \quad (14)$$

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

$$\mathbf{d}_t = (1-\rho_t)\mathbf{d}_{t-1} + \rho_t \mathbf{g}(\mathbf{x}_t, \mathbf{y}_t)$$
$$\mathbf{v}_t = \operatorname{argmin}_{\mathbf{v}\in\mathcal{C}}\langle \mathbf{d}_t, \mathbf{v}\rangle$$
$$\mathbf{x}_{t+1} = (1-\gamma_{t+1})\mathbf{x}_t + \gamma_{t+1}\mathbf{v}_t. \quad (15)$$

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 at each time followed by averaging, i.e., $\{\mathbf{z}_{i,t}\}_{i=1}^m$ for which we have,

$$\mathbf{g}_m(\mathbf{x}_t; \mathbf{y}_t, \mathbf{z}_{i,t})$$
$$= \frac{1}{m}\sum_{i=1}^{m}\left(\frac{F(\mathbf{x}_t + c_t\mathbf{z}_{i,t}; \mathbf{y}) - F(\mathbf{x}_t; \mathbf{y})}{c_t}\mathbf{z}_{i,t}\right). \quad (16)$$

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 [11] for RDSA,

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

Using (17), similar statistics for the improvised RDSA gradient estimator can be evaluated as follows:

$$\mathbb{E}_{\mathbf{z}_t\sim\mu, \mathbf{y}_t\sim\mathcal{P}}\left[\mathbf{g}_m(\mathbf{x}; \mathbf{y}_t, \mathbf{z}_t)\right] = \nabla f(\mathbf{x}) + \frac{c_t}{m} L \mathbf{v}(\mathbf{x}, c_t)$$
$$\mathbb{E}_{\mathbf{z}_t\sim\mu, \mathbf{y}_t\sim\mathcal{P}}\left[\|\mathbf{g}_m(\mathbf{x}; \mathbf{y}_t, \mathbf{z}_t)\|^2\right] \leq \left(\frac{1+m}{2m}\right) c_t^2 L^2 M(\mu)$$
$$+ 2\left(1 + \frac{s(d)}{m}\right)\mathbb{E}\left[\|\nabla F(\mathbf{x}; \mathbf{y}_t)\|^2\right], \quad (18)$$

where $\|\mathbf{v}(\mathbf{x}, c_t)\| \leq \frac{1}{2}\mathbb{E}\left[\|\mathbf{z}\|^3\right]$. A proof for (18) can be found in [30]. 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 state the zeroth order stochastic Frank-Wolfe algorithm which is presented in algorithm 2.

5. Decentralized Stochastic Constrained Zeroth Order Optimization

For a decentralized setting, the network architecture typically consists of workers and a master node. We specifically assume there are M workers and one master node. In this scenario, we assume that the data is distributed across all the workers which in turn are all drawn from possibly different data distributions

Algorithm 2 Decentralized Stochastic Gradient Free Frank Wolfe

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**

and the workers try to solve the optimization problem in (1). In essence, we address the following problem

$$\min_{\mathbf{x}\in\mathcal{C}} \left\{ f(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^M f_i(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^M \mathbb{E}_{\mathbf{y}\sim\mathcal{P}_i}\left[F(\mathbf{x};\mathbf{y})\right] \right\}, \tag{19}$$

where $\mathcal{P}_i$ is the local data distribution at node i.
We specifically employ Algorithm 2 to solve the optimization problem in (29) 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 non-convex loss functions is based on the evolution of the mean square error of the surrogate gradient term that we track through the gradient averaging technique. We quantify the primal gap for convex loss functions next.

Theorem 5.1 (Convex Losses [47])**.** *Let Assumptions A1-A4 hold and let each f_i be L-smooth. Let the sequence γ_t be given by $\gamma_t = \frac{2}{t+8}$. In order to achieve a primal sub-optimality gap of ϵ for convex loss functions in Algorithm 2,i.e.,*

$$\mathbb{E}\left[f(\mathbf{x}_t) - f(\mathbf{x}^*)\right] \leq \epsilon, \tag{20}$$

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

We provide a brief proof sketch of Theorem 5.1. Theorem 5.1 establishes the dimension dependence of the primal gap to be $d^{1/3}$. The iteration dependence, i.e., $O(T^{-1/3})$ matches that of the stochastic Frank-Wolfe with first order information as in [36]. The dimension dependence for the number of queries cannot be improved as it matches the minimax lower bound in [11]. Denote the Frank-Wolfe 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 Lemma 3.3 from [47], which characterizes the evolution of the primal gap for the function and the main result of the Lemma can be stated as follows:

$$f(\mathbf{x}_{t+1}) - f(\mathbf{x}^*) \leq (1-\gamma_{t+1})(f(\mathbf{x}_t) - f(\mathbf{x}^*)) + \gamma_{t+1}R\|\nabla f(\mathbf{x}_t) - \mathbf{g}_t\| + \frac{LR^2\gamma_{t+1}^2}{2}. \tag{21}$$

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

$$\begin{aligned}\mathbb{E}\left[\|\nabla f_i(\mathbf{x}_t) - \mathbf{g}_{i,t}\|^2\right] &\leq 2\rho_t^2\left(\sigma^2 + 2L_1^2\right) + \frac{\rho_t}{2m^2}c_t^2L^2M(\mu) + 8\rho_t^2\left(1 + \frac{s(d)}{m}\right)L_1^2 \\ &+ \left(\frac{1+m}{2m}\right)\rho_t^2c_t^2L^2M(\mu) + \frac{2L^2R^2\gamma_t^2}{\rho_t} \\ &+ \left(1 - \frac{\rho_t}{2}\right)\mathbb{E}\left[\|\nabla f_i(\mathbf{x}_{t-1}) - \mathbf{g}_{i,t-1}\|^2\right],\end{aligned} \tag{22}$$

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 the following Lemma from [47],

Lemma 5.2. *Let $z(k)$ be a non-negative (deterministic) sequence satisfying:*

$$z(k+1) \leq (1 - r_1(k))\, z_1(k) + r_2(k),$$

where $\{r_1(k)\}$ and $\{r_2(k)\}$ are deterministic sequences with

$$\frac{a_1}{(k+1)^{\delta_1}} \leq r_1(k) \leq 1 \text{ and } r_2(k) \leq \frac{a_2}{(k+1)^{2\delta_1}},$$

with $a_1 > 0$, $a_2 > 0$, $1 > \delta_1 > 1/2$ and $k_0 \geq 1$. Then,

$$z(k+1) \leq \exp\left(-\frac{a_1\delta_1(k+k_0)^{1-\delta_1}}{4(1-\delta_1)}\right)\left(z(0) + \frac{a_2}{k_0^{\delta_1}(2\delta_1 - 1)}\right) + \frac{a_2 2^{\delta_1}}{a_1\left(k+k_0\right)^{\delta_1}}.$$

Using Lemma 5.2 above, we obtain,

$$\mathbb{E}\left[\|\nabla f(\mathbf{x}_t) - \mathbf{g}_t\|^2\right] \leq \frac{2Q}{(t+8)^{2/3}}, \tag{23}$$

where Q is a constant characterized by the different algorithm parameters L, R, σ^2, d and m. Plugging (23) to (21) provides the result needed.
For convex losses, the dual gap is given by,

$$\mathbb{E}\left[\min_{t=0,\cdots,T-1}\mathcal{G}(\mathbf{x}_t)\right] \leq O\left(T^{-1/3}\right). \tag{24}$$

The duality gap characterization is obtained by using the following inequality for the duality gap from [47]:

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

where for $\mathbb{E}\left[\|\nabla f(\mathbf{x}_t) - \mathbf{g}_t\|\right]$ we use the relation in (23).
Technically speaking, while it is important to characterize the primal gap to benchmark the performance, it is important to characterize the dual gap as it is used as a stopping criterion for Frank-Wolfe algorithms.
We quantify the performance of the algorithm for non-convex loss functions using Frank-Wolfe duality gap. The usage of Frank-Wolfe duality gap to characterize the convergence for non-convex loss functions is standard in literature (see, for example [44]. 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 5.3 (Non-Convex Losses [47])**.** *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 dual gap of ϵ for non-convex loss functions in Algorithm 2 ,i.e.,*

$$\mathbb{E}\left[\min_{t=0,\cdots,T-1} \mathcal{G}(\mathbf{x}_t)\right] \leq \epsilon, \tag{25}$$

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

Theorem 5.3 establishes the dimension dependence of the Frank-Wolfe duality gap for non-convex losses to be $d^{1/3}$ and has the same iteration dependence i.e., $O(T^{-1/4})$ as that of SFW in [44].
We briefly sketch the main proof idea of Theorem 5.3. The proof for Theorem 5.3 follows similarly by using the following inequality for the duality gap from [47]:

$$\begin{aligned}\gamma\mathbb{E}\left[\mathcal{G}(\mathbf{x}_t)\right] &\leq \mathbb{E}\left[f(\mathbf{x}_t)\right] - \mathbb{E}\left[f(\mathbf{x}_{t+1})\right] \\ &+ \gamma R\mathbb{E}\left[\|\nabla f(\mathbf{x}_t) - \mathbf{g}_t\|\right] + \frac{LR^2\gamma^2}{2}.\end{aligned} \tag{26}$$

For each f_i, we have that,

$$\mathbb{E}\left[\|\nabla f_i(\mathbf{x}_t) - \mathbf{g}_{i,t}\|^2\right] = O\left(\frac{(d/m)^{2/3}}{(t+9)^{1/2}}\right), \forall\ t = 0, \cdots, T-1 \tag{27}$$

and, thus,

$$\begin{aligned}\mathbb{E}\left[\|\nabla f(\mathbf{x}_t) - \mathbf{g}_t\|^2\right] &\leq \frac{1}{M}\sum_{i=1}^{M}\mathbb{E}\left[\|\nabla f_i(\mathbf{x}_t) - \mathbf{g}_{i,t}\|^2\right] \\ &= O\left(\frac{(d/m)^{2/3}}{(t+9)^{1/2}}\right), \forall\ t = 0, \cdots, T-1\end{aligned}$$

Using the bound derived in (23) and then summing both sides of (26), we have that,

$$\begin{aligned}\mathbb{E}\left[\min_{t=0,\cdots,T-1} \mathcal{G}(\mathbf{x}_t)\right] &\leq \frac{\mathbb{E}\left[f(\mathbf{x}_0)\right] - \mathbb{E}\left[f(\mathbf{x}^*)\right]}{T^{1/4}} \\ &+ \frac{Q_{nc}Rd^{1/3}}{T^{1/4}m^{1/3}} + \frac{LR^2}{2T},\end{aligned}$$

where Q_{nc} is a constant summarizing the dependence of other algorithm parameters. The stated result is thus established.
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 mean square error (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 the next part of this paper, where we use SPIDER [12], a variance reduction technique to further improve the performance.

6. Decentralized Variance Reduced Stochastic Frank-Wolfe

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):

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

where each function i is available to the i-th 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, i.e., the finite sum optimization problem in (28) can be further written as:

$$\min_{\mathbf{x}\in\mathcal{C}} \left\{ f(x) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}) = \sum_{i=1}^{M}\sum_{j=1}^{n} f_{i,j}(\mathbf{x}) \right\}. \tag{29}$$

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, [12] proposed Stochastic Path-Integrated Differential Estimator (SPIDER) technique, for unconstrained optimization in centralized settings for both first order and zeroth order oracles. In this paper, we generalize SPIDER to the constrained, distributed and zeroth order settings.

A. SPIDER

Stochastic Path-Integrated Differential EstimatoR, or SPIDER is built for dynamic tracking, while avoiding excessive querying to oracles and ultimately reduce query complexity. We provide a brief overview of SPIDER before proceeding further and refer the reader to [12] 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

Algorithm 3 Decentralized Variance Reduced Zeroth Order Frank Wolfe

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

Output: $\mathbf{x}_T$.

Initialize $\mathbf{x}_0 \in \mathcal{C}$

for $t = 0, 2, \ldots, T-1$ **do**

  At each worker i compute

  if then mod(t,q)=0

    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 $\mathbf{e}_j$.

    Each worker updates $\mathbf{g}_{i,t} = \mathbf{g}_i(\mathbf{x}_t)$

  else

    Draw S_2 pairs of component functions and Gaussian random vectors $\{\mathbf{z}\}$ at each worker i and update

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

    Each worker updates $\mathbf{g}_{i,t} = \mathbf{g}_i(\mathbf{x}_t) + \mathbf{g}_{i,t-1}$

  end if

  Each worker pushes $\mathbf{g}_{i,t}$ to the master node.

  Master node computes $\mathbf{g}_t = \frac{1}{M}\sum_{i=1}^{M} \mathbf{g}_{i,t}$ and sends it to all workers.

  Master node computes $\mathbf{v}_t = \operatorname{argmin}_{\mathbf{s}\in\mathcal{C}}\langle \mathbf{s}, \mathbf{g}_t\rangle$,

  Master node computes $\mathbf{x}_{t+1} = (1-\gamma_t)\mathbf{x}_t + \gamma_t \mathbf{v}_t$ and sends it to all workers.

end for

of each period, i.e., $mod(t,q) = 0$, 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 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, 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 Frank Wolfe is outlined in Algorithm 3.

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 mean square error in terms of ϵ primal gap for convex losses and ϵ Frank-Wolfe duality gap for non-convex losses. The major improvement of the variance reduction scheme is in terms of the gradient tracking performance, i.e., with $S_2 = \frac{(2d+9)\sqrt{n}}{n_0}$, $q = \frac{n_0\sqrt{n}}{6}$, where $n_0 \in \left[1, \frac{\sqrt{n}}{6}\right]$, the mean squared error of the tracked gradient error is given by:

$$\mathbb{E}\left[\|\nabla f(\mathbf{x}_t) - \mathbf{g}_t\|^2\right] \leq \frac{\epsilon^2}{3}. \tag{30}$$

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

Theorem 6.1. *Let Assumptions A1-A4 hold and let each. Consider algorithm 3 with* $S_2 = \frac{(2d+9)\sqrt{Mn}}{n_0}$, $q = \frac{n_0\sqrt{Mn}}{6}$, *where* $n_0 \in \left[1, \frac{\sqrt{Mn}}{6}\right]$, $\gamma_t = O(\epsilon)$ *and* $S_1 = Mnd$, *we have that the number of queries required to obtain a Frank-Wolfe duality gap of* ϵ, *i.e.,*

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

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

The proof of Theorem 6.1 follows similarly as that of the proof of theorem 5.3 and the gradient approximation bound in (30) is as derived in Lemma 11 of [12]. For a fair comparison with the stochastic case, the query complexity obtained for the finite-sum case can be reproduced by $S_2 = \frac{30(2d+9)\sigma}{n_0\epsilon}$, $q = \frac{5n_0\sigma}{\epsilon}$, where $n_0 \in \left[1, \frac{\sqrt{Mn}}{6}\right]$, $\gamma_t = O(\epsilon)$ and $S_1 = \frac{96d\sigma^2}{\epsilon^2}$. Theorem 6.1 improves the iteration dependence even further from that of the vanilla decentralized stochastic Frank-Wolfe in the regime where $Mn \geq \epsilon^{-2}$. However, the dimension dependence can 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.

7. Distributed Zeroth Order Frank Wolfe

In this section, we study the finite-sum version of the optimization problem at hand, i.e.,

$$\min_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x}) = \min_{\mathbf{x}\in\mathcal{C}} \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}), \tag{31}$$

in a network of M agents, where the i-th agent has access to the i-th function. In this section, we consider a fully distributed setup which is devoid of a central coordinator. In order to exchange information, the nodes use peer-to-peer message passing while conforming to pre-specified 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 linear minimization oracle. The inter-node communication network to which the information exchange between nodes conforms to is modeled as an *undirected* simple connected graph $G = (V, E)$, with $V = [1 \cdots M]$ 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, i.e., $\bar{\mathbf{x}}_t^i$, 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,

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

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, i.e.,

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

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

$$\bar{\mathbf{g}}_t^i \leftarrow \sum_{j=1}^{M} W_{ij}\mathbf{G}_t^j \tag{32}$$

Finally, each node i undergoes the Frank-Wolfe step:

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

It is worth noting that unlike other distributed optimization protocols such as distributed SGD which involve just one round of communication, a distributed Frank-Wolfe algorithm requires two rounds of communication once for the iterate and once 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. 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.

Algorithm 4 Distributed Zeroth Order Frank-Wolfe

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 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**: $\bar{\mathbf{x}}_{t+1}^i, \forall\ i \in [M]$.

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 inter-agent communication graph is connected, i.e., $\lambda_2(\mathbf{L}) > 0$.

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 [51], where the authors study a distributed first order Frank-Wolfe 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

$$\max_{i\in[M]} \|\bar{\mathbf{x}}_t^i - \bar{\mathbf{x}}_t\|_2 = O\left(\frac{1}{t^\alpha}\right),\ \forall\ t \geq 1, \tag{33}$$

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,

$$\max_{i\in[M]} \|\bar{\mathbf{g}}_t^i - \bar{\mathbf{g}}_t\|_2 = O\left(\frac{1}{t^\alpha}\right),\ \forall\ t \geq 1, \tag{34}$$

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 depend 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 (32) 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 7.1 (Convex Losses)**.** *Let Assumptions A1-A6 hold and let each f_i be L-smooth. Let the sequence γ_t be given by $\gamma_t = \frac{2}{t+1}$. In order to achieve a primal sub-optimality gap of ϵ for convex loss functions in Algorithm 4, i.e.,*

$$f(\bar{\mathbf{x}}_t) - f(\mathbf{x}^*) \leq \epsilon, \tag{35}$$

the number of queries to the IZO is given by $O\left(\frac{d}{\epsilon}\right)$.

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,

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

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

$$\begin{aligned}
&\langle \mathbf{v}_t^i - \bar{\mathbf{x}}_t, \nabla f(\bar{\mathbf{x}}_t)\rangle \leq \langle \mathbf{v}_t^i - \bar{\mathbf{x}}_t, \bar{\mathbf{g}}_t^i\rangle + R\left\|\bar{\mathbf{g}}_t^i - \nabla f(\bar{\mathbf{x}}_t)\right\| \\
&\leq \langle \mathbf{v} - \bar{\mathbf{x}}_t, \bar{\mathbf{g}}_t^i\rangle + R\left\|\bar{\mathbf{g}}_t^i - \nabla f(\bar{\mathbf{x}}_t)\right\|,\ \forall v \in \mathcal{C} \\
&\leq \langle \mathbf{v} - \bar{\mathbf{x}}_t, \nabla f(\bar{\mathbf{x}}_t)\rangle + 2R\left\|\bar{\mathbf{g}}_t^i - \nabla f(\bar{\mathbf{x}}_t)\right\|,\ \forall v \in \mathcal{C}, \quad (37)
\end{aligned}$$

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,

$$\begin{aligned}
\left\|\bar{\mathbf{g}}_t^i - \nabla f(\bar{\mathbf{x}}_t)\right\| &\leq \left\|\bar{\mathbf{g}}_t - \frac{1}{M}\sum_{i=1}^{M}\nabla f_i(\bar{\mathbf{x}}_t^i)\right\| + \left\|\bar{\mathbf{g}}_t^i - \bar{\mathbf{g}}_t\right\| \\
&+ \left\|\frac{1}{M}\sum_{i=1}^{M}\nabla f_i(\bar{\mathbf{x}}_t^i) - \nabla f(\bar{\mathbf{x}}_t)\right\|, \quad (38)
\end{aligned}$$

where for the 2nd and 3rd terms we use the bound derived in (34), where we use $\alpha = 1$. For the 1st 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 algorithm parameters, i.e., L, R and M. Substituting back into (36) 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,

$$h_{t+1} \leq h_t + \gamma_t\langle \bar{\mathbf{v}}_t - \bar{\mathbf{x}}_t, \nabla f(\bar{\mathbf{x}}_t)\rangle + c_2\gamma_t^2. \tag{39}$$

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 (39) 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 7.2 (Non-Convex 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 $\gamma_t = \frac{1}{\sqrt{t}}$. Then, in order to obtain a dual gap of ϵ for non-convex loss functions in Algorithm 4, i.e.,*

$$\min_{t=\lfloor T/2\rfloor+1,\cdots,T}\mathcal{G}\left(\bar{\mathbf{x}}_t\right) \leq \epsilon, \tag{40}$$

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

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

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

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 (37) and (38) for $\alpha = 1/2$, we have that,

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

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,

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

Combining the above inequalities, yields

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

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

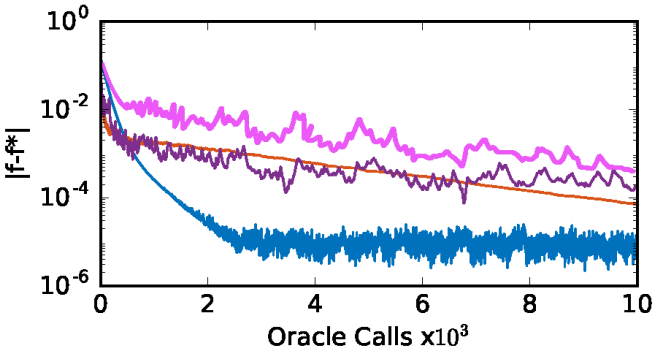


Fig. 1: Covtype Dataset: Summary of comparison between 1st order PGD (blue), 1st order FW (red), decentralized stochastic 0th order FW (maroon) and distributed 0th order FW (pink).

It is worth noting that the problem being considered in this section is a deterministic version of Frank-Wolfe and hence a better query complexity is obtained in comparison to stochastic Frank-Wolfe 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 are in the order sense and specifically explicitly quantify the dimension dependence and the iteration dependence.

8. 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 dataset 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 1st order PGD and 1st order Frank-Wolfe. 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 Frank-Wolfe based algorithms pay a price for the projection free nature of the algorithm and are outperformed by 1st order PGD. 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 Frank-Wolfe 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. Non-Convex 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, i.e., 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 pre-specified 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:

$$\mathbf{x}_{\text{adv}} = \operatorname*{argmax}_{\mathbf{x}' : \|\mathbf{x}' - \mathbf{x}\|_p \leq \epsilon_p} L(\mathbf{x}', y). \tag{41}$$

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 [15] for a detailed treatment), Projected Gradient Descent (PGD) have been particularly effective in generating adversarial examples for state-of-the-art deep networks built for challenging computer vision datasets. 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 only access 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 [20], [21], [37]. 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 a example, whereas in decision-based attacks, the attacker only gets to observe the hard label predicted by the classifier. Decision-based attacks have been studied by [2], [4], [5], 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 [41]. A real-world application of black-box attacks to fool a PDF malware classifier was demonstrated by [53], for which a genetic algorithm was used. The paper [39] 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 [5], [20]. Though [5] generated successful attacks attaining high attack accuracy, the method was found to be extremely query hungry which was then remedied to an extent by [20]. In [21], the authors exploit correlation of gradients across iterations by setting a prior and use a piece wise constant perturbation, i.e., tiling to develop a query efficient black-box method. In addition to that, the method developed in [21], is essentially a stochastic Frank-Wolfe 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* [38] which can be formalized as follows as a constrained stochastic optimization problem:

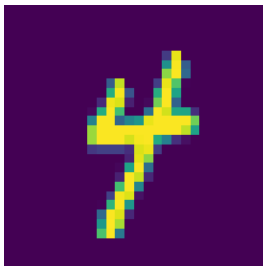
$$\boldsymbol{\delta}' = \underset{\|\boldsymbol{\delta}\|_p \leq \epsilon_p}{\operatorname{argmax}} \mathbb{E}_{(\mathbf{x},y)\in(\mathcal{X},\mathcal{Y})} \left[L(\mathbf{x}+\boldsymbol{\delta}, y) \right]. \tag{42}$$

Given the need to take a pass over possibly high-dimensional data for the optimization problem at hand in (42) and the huge dimension and size of computer vision datasets such as MNIST [28] 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 Frank-Wolfe which we demonstrate below. A crucial point to note here is that, it is not necessary to find the global maximizer in (42). Instead it is just enough to find a "good enough" feasible point in the constraint set which misclassifies most images in the set.

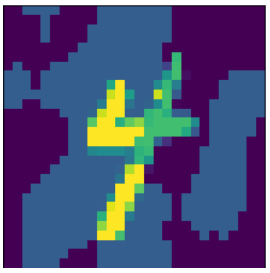
We perform an experiment on MNIST to find a universal adversarial perturbation. We use the pre-trained LeNet-5 model available from Pytorch to demonstrate the attacks. We enforce the information access limited to the softmax layer, i.e, only score function values can be obtained for different classes after feeding an input image. We set the constraint to be that of ℓ_∞ with $\epsilon_p = 0.25$ in (42). 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 Frank-Wolfe 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 10 worker nodes with 100 images each, with 10 images drawn from each class. For the distributed setting, we also consider a network of 10 nodes with 100 images each, with 10 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 $\boldsymbol{\delta}$'s obtained at each of the nodes for the distributed setting. Figures 2-4 provide some examples of the universal adversarial perturbation generated. Table I summarizes the performance of a distributed stochastic FW and benchmarks it with respect to the decentralized FW. As it can be seen from Table I, the performance of the distributed scheme in terms of the attack success rate is very close to that of its decentralized counterpart.

TABLE I: Summary of ℓ_∞ universal adversarial perturbation with $\epsilon = 0.25$ MNIST attacks using Decentralized and Distributed Frank-Wolfe (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%



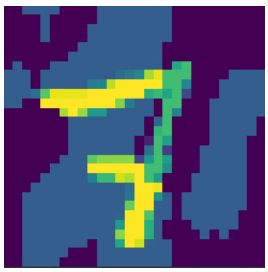
(a) Original Image: Class 4



(b) Perturbed Image: Class 8

Fig. 2: An image of 4 changed to 8 with the adversarial perturbation

(a) Original Image: Class 7



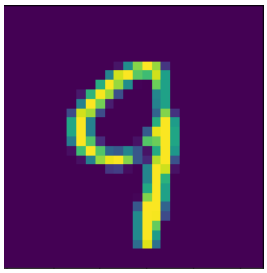
(b) Perturbed Image: Class 3

Fig. 3: An image of 7 changed to 3 with the adversarial perturbation

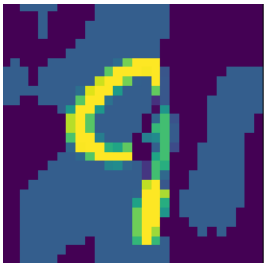
9. Conclusion

In this paper, 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 stochastic Frank-Wolfe frameworks for distributed and decentralized constrained stochastic optimization. We have illustrated quantitative aspects of the performance of stochastic Frank-Wolfe framework 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 Frank-Wolfe frameworks for black-box adversarial attacks in the context of deep neural networks.

Anit Kumar Sahu (S'13, M'18) received a B.Tech. degree in electronics and electrical communication engineering, and a M.Tech. degree in telecommunication systems engineering from the Indian Institute of Technology Kharagpur, India, in May 2013 and a Ph.D. in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA in December 2018. Since January 2019, he has been with the Bosch Center for Artificial Intelligence as a machine learning research scientist. His research interests include stochastic optimization, robust machine learning and distributed inference in large-scale systems. He is a recipient of the A.G. Jordan Award from the Department of Electrical and Computer Engineering at Carnegie Mellon University.



(a) Original Image: Class 9



(b) Perturbed Image: Class 8

Fig. 4: An image of 9 changed to 8 with the adversarial perturbation

Soummya Kar (S'05, M'10) received a B.Tech. in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, in May 2005 and a Ph.D. in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 2010. From June 2010 to May 2011, he was with the Electrical Engineering Department, Princeton University, Princeton, NJ, USA, as a Postdoctoral Research Associate. He is currently an Associate Professor of Electrical and Computer Engineering at Carnegie Mellon University, Pittsburgh, PA, USA. His research interests include decision-making in large-scale networked systems, stochastic systems, multi-agent systems and data science, with applications in cyber-physical systems and smart energy systems.

References

[1] Krishnakumar Balasubramanian and Saeed Ghadimi. Zeroth-order (non)-convex stochastic optimization via conditional gradient and gradient updates. In *Advances in Neural Information Processing Systems*, pages 3459–3468, 2018.

[2] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

[3] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[4] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. 2019.

[5] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, AISec '17, pages 15–26, New York, NY, USA, 2017. ACM.

[6] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26. ACM, 2017.

[7] Xiangyi Chen, Sijia Liu, Kaidi Xu, Xingguo Li, Xue Lin, Mingyi Hong, and David Cox. Zo-adamm: Zeroth-order adaptive momentum method for black-box optimization. In *Advances in Neural Information Processing Systems*, pages 7202–7213, 2019.

[8] Krzysztof Choromanski, Mark Rowland, Vikas Sindhwani, Richard E Turner, and Adrian Weller. Structured evolution with compact architectures for scalable policy optimization. *arXiv preprint arXiv:1804.02395*, 2018.

[9] F. R. K. Chung. *Spectral Graph Theory*. Providence, RI : American Mathematical Society, 1997.

[10] Stanley N Deming, Lloyd R Parker Jr, and M Bonner Denton. A review of simplex optimization in analytical chemistry. *CRC Critical Reviews in Analytical Chemistry*, 7(3):187–202, 1978.

[11] John C Duchi, Michael I Jordan, Martin J Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

[12] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems*, pages 689–699, 2018.

[13] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[14] Dan Garber and Elad Hazan. Faster rates for the frank-wolfe method over strongly-convex sets. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 541–549. JMLR. org, 2015.

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[16] Genetha Anne Gray, Tamara G Kolda, Ken Sale, and Malin M Young. Optimizing an empirical scoring function for transmembrane protein structure determination. *INFORMS Journal on Computing*, 16(4):406–418, 2004.

[17] Davood Hajinezhad, Mingyi Hong, and Alfredo Garcia. Zeroth order nonconvex multi-agent optimization over networks. *arXiv preprint arXiv:1710.09997*, 2017.

[18] Elad Hazan and Satyen Kale. Projection-free online learning. In *International Conference on Machine Learning*, pages 1843–1850, 2012.

[19] Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. In *International Conference on Machine Learning*, pages 1263–1271, 2016.

[20] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2137–2146, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[21] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. Prior convictions: Black-box adversarial attacks with bandits and priors. In *International Conference on Learning Representations*, 2019.

[22] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *ICML (1)*, pages 427–435, 2013.

[23] Kaiyi Ji, Zhe Wang, Yi Zhou, and Yingbin Liang. Improved zeroth-order variance reduced algorithms and analysis for nonconvex optimization. In *International Conference on Machine Learning*, pages 3100–3109, 2019.

[24] S. Kar, J. M. F. Moura, and K. Ramanan. Distributed parameter estimation in sensor networks: nonlinear observation models and imperfect communication. *IEEE Transactions on Information Theory*, 58(6):3575 – 3605, June 2012.

[25] Jack Kiefer, Jacob Wolfowitz, et al. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[26] Simon Lacoste-Julien. Convergence rate of frank-wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345*, 2016.

[27] Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *Advances in Neural Information Processing Systems*, pages 496–504, 2015.

[28] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[29] Sijia Liu, Jie Chen, Pin-Yu Chen, and Alfred Hero. Zeroth-order online alternating direction method of multipliers: Convergence analysis and applications. In *International Conference on Artificial Intelligence and Statistics*, pages 288–297, 2018.

[30] Sijia Liu, Jie Chen, Pin-Yu Chen, and Alfred Hero. Zeroth-order online alternating direction method of multipliers: Convergence analysis and applications. In *International Conference on Artificial Intelligence and Statistics*, pages 288–297, 2018.

[31] Sijia Liu, Bhavya Kailkhura, Pin-Yu Chen, Paishun Ting, Shiyu Chang, and Lisa Amini. Zeroth-order stochastic variance reduction for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 3727–3737, 2018.

[32] C. G. Lopes and A. H. Sayed. Diffusion least-mean squares over adaptive networks: Formulation and performance analysis. *IEEE Transactions on Signal Processing*, 56(7):3122–3136, July 2008.

[33] Dhruv Malik, Ashwin Pananjady, Kush Bhatia, Koulik Khamaru, Peter L Bartlett, and Martin J Wainwright. Derivative-free methods for policy optimization: Guarantees for linear quadratic systems. *arXiv preprint arXiv:1812.08305*, 2018.

[34] Alison L Marsden, Jeffrey A Feinstein, and Charles A Taylor. A computational framework for derivative-free optimization of cardiovascular geometries. *Computer methods in applied mechanics and engineering*, 197(21-24):1890–1905, 2008.

[35] Alison L Marsden, Meng Wang, JE Dennis, and Parviz Moin. Trailing-edge noise reduction using derivative-free optimization and large-eddy simulation. *Journal of Fluid Mechanics*, 572:13–36, 2007.

[36] Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. Conditional gradient method for stochastic submodular maximization: Closing the gap. In *International Conference on Artificial Intelligence and Statistics*, pages 1886–1895, 2018.

[37] Seungyong Moon, Gaon An, and Hyun Oh Song. Parsimonious black-box adversarial attacks via efficient combinatorial optimization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4636–4645, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[38] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

[39] Nina Narodytska and Shiva Kasiviswanathan. Simple black-box adversarial attacks on deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1310–1318. IEEE, 2017.

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

[41] Blaine Nelson, Benjamin IP Rubinstein, Ling Huang, Anthony D Joseph, Steven J Lee, Satish Rao, and JD Tygar. Query strategies for evading convex-inducing classifiers. *Journal of Machine Learning Research*, 13(May):1293–1332, 2012.

[42] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2004.

[43] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. Technical report, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2011.

[44] Sashank J Reddi, Suvrit Sra, Barnabás Póczos, and Alex Smola. Stochastic frank-wolfe methods for nonconvex optimization. In *Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on*, pages 1244–1251. IEEE, 2016.

[45] Andrzej Ruszczyński. A merit function approach to the subgradient method with averaging. *Optimisation Methods and Software*, 23(1):161–172, 2008.

[46] Anit Kumar Sahu, Dusan Jakovetic, Dragana Bajovic, and Soummya Kar. Distributed zeroth order optimization over random networks: A Kiefer-Wolfowitz stochastic approximation approach. In *57th IEEE Conference on Decision and Control (CDC)*, Miami, 2018.

[47] Anit Kumar Sahu, Manzil Zaheer, and Soummya Kar. Towards gradient free and projection free stochastic optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3468–3477, 2019.

[48] Zebang Shen, Cong Fang, Peilin Zhao, Junzhou Huang, and Hui Qian. Complexities in projection-free stochastic non-convex minimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2868–2876, 2019.

[49] Yujie Tang and Na Li. Distributed zero-order algorithms for nonconvex multi-agent optimization. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 781–786. IEEE, 2019.

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

[51] Hoi-To Wai, Jean Lafond, Anna Scaglione, and Eric Moulines. Decentralized frank–wolfe algorithm for convex and nonconvex problems. *IEEE Transactions on Automatic Control*, 62(11):5522–5537, 2017.

[52] Yining Wang, Simon Du, Sivaraman Balakrishnan, and Aarti Singh. Stochastic zeroth-order optimization in high dimensions. In *International Conference on Artificial Intelligence and Statistics*, pages 1356–1365, 2018.

[53] Weilin Xu, Yanjun Qi, and David Evans. Automatically evading classifiers. In *Proceedings of the 2016 Network and Distributed Systems Symposium*, pages 21–24, 2016.

[54] Alp Yurtsever, Suvrit Sra, and Volkan Cevher. Conditional gradient methods via stochastic path-integrated differential estimator. In *Proceedings of the International Conference on Machine Learning-ICML 2019*, number CONF, 2019.